MISCELLANEOUS PAPER O-79-4

PRIMER ON COMPUTER GRAPHICS PROGRAMMING

by

Westinghouse Word Processing Center
Pittsburgh, Pa. 15230

October 1979
Final Report

Prepared for Office, Chief of Engineers, U. S. Army
Washington, D. C. 20314

Under Contract No. DACW39-78-M-2676

Monitored by Automatic Data Processing Center
U. S. Army Engineer Waterways Experiment Station
P. O. Box 631, Vicksburg, Miss. 39180

Destroy this report when no longer needed. Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Miscellaenous Paper O-79-4	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* PRIMER ON COMPUTER GRAPHICS PROGRAMMING		5. TYPE OF REPORT & PERIOD COVERED Final report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)*		8. CONTRACT OR GRANT NUMBER*(s)* Contract No. DACW39-78-M-2676
9. PERFORMING ORGANIZATION NAME AND ADDRESS Westinghouse Word Processing Center Pittsburgh, Pa. 15230		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office, Chief of Engineers, U. S. Army Washington, D. C. 20314		12. REPORT DATE October 1979 13. NUMBER OF PAGES 201
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* U. S. Army Engineer Waterways Experiment Station Automatic Data Processing Center P. O. Box 631, Vicksburg, Miss. 39180		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT *(of this Report)* Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)* Computer graphics Graphics Compatibility System Computer programming		

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report is a tutorial for learning graphics programming with the Graphics Compatibility System (GCS). GCS is a collection of ANSI standard FORTRAN subroutines invocable by a user's FORTRAN program. The report covers GCS's capabilities. GCS can produce a simple line drawing or handle comprehensive general purpose axis creation and automatic clipped; arc and conic generation; graphics input; secondary coordinate systems; multiple software character fonts; data structures; segmentation; and color. The report addresses both two- and three-dimensional graphics capability within GCS.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

PREFACE

This report is the result of work performed under Contract No. DACW39-78-M-2676, dated 3 April 1978, between the U. S. Army Engineer Waterways Experiment Station (WES), Vicksburg, Miss., and the Westinghouse Word Processing Center, Pittsburgh, Pa. The work concerned publishing a word processing version of the Primer on Computer Graphics Programming. The original version of this primer was developed at the U. S. Military Academy.

Contributors to the document were Dr. Richard Puk, Sandia Laboratories; Dr. Steve Orbon and Mr. Robert Bruns, Westinghouse; and Mr. James M. Jones II, R&D Software Group, Automatic Data Processing (ADP) Center, WES. The work was administered by the ADP Center, WES, as part of Computer Technology-Engineering Software, Project No. 4A762725AT11, and the Civil Works Computation and Analysis Project sponsored by the Office, Chief of Engineers, U. S. Army (OCE). OCE points of contact were Mr. Robert McMurrer (DAEN-DSE) and Messrs. Richard Malm and Harry Hardin (DAEN-CWE-BA).

Mr. Jones monitored the contract under the general supervision of Dr. N. Radhakrishnan, Special Technical Assistant to the Chief of the ADP Center, WES, and Mr. D. L. Neumann, Chief of the ADP Center.

Directors of WES during the period of the contract and the preparation of this report were COL J. L. Cannon, CE, and COL N. P. Conover, CE. Technical Director was Mr. F. R. Brown.

CONTENTS

INTRODUCTION

THE GCS RESEARCH AND DEVELOPMENT PROJECT

The traditional forms of computer output — vast piles of densely printed computer listings — often leave managers, engineers and students confused and frustrated. Often the information one wants to know is somewhere in the computer output, but not in a form wherein the user can find what he wants, understands, or can easily assimilate.

For years the computer field has looked forward to breaking this bottleneck in computer-human communications by using pictures instead of printouts. The idea is not new. The first stored-program, electronic computer ever designed had in its initial design an oscilliscope which was supposed to give graphic displays of the solutions of the problems it was to solve. Working two-way man-to-computer-to-man graphic communications dates back at least 10-20 years to the early Cape Cod and SAGE Air Defense Systems and Ivan Sutherland's early SKETCHPAD system. Yet, even today, the amount of graphic output that most computer users ever see is limited to a few crude barcharts of graphs produced on computer printers, or possibly a pen-plot made with an electromechanical plotter. Everyone knows about computer graphics — but very few actually have it, and even fewer make good use of it in their normal day-to-day work.

Computer graphics has been too expensive. The hardware has been expensive but, far worse than that, the software has been actrociously expensive. It has been expensive not only in initial cost, but also in its tendency to be both highly specialized and highly machine-dependent. Only specially qualified assembly language programmers could work on it and it would work on only one machine. Obviously there were exceptions to this. Much of the great acceptance with pen-plotters received in recent years came from the fact that they permitted the use of FORTRAN subroutines which could be used by run-of-the-mill FORTRAN programmers to plot simple axes, graphs, etc. The more recent success of the TEKTRONIX storage tube terminals has also been greatly influenced by the fact that reasonably good user-accessible software is available for such devices.

Today we are in a period of rapid improvement in availability of practical computer graphics. Timesharing is bringing the computer out of the warrens of the computer center and into the offices of engineers, managers, classrooms, academic buildings, and even the dormitories of academia. Timesharing started out with printing terminals, like the teletype but the era of timesharing graphics is fast approaching. LSI and MOS technology are making the minicomputer and even the microcomputer increasingly attractive. Minicomputer-based special purpose graphics systems are now very successfully paying their own way in a number of tasks such as in the layout of electronic circuit boards. Coupled with small mass storage devices such as discs, or even in some cases cassettes, minicomputer-based graphics systems have considerable capability, in spite of serious software limitations. Today they seem ready to burst out into widespread use. Perhaps the most exciting possibility of all is the intelligent terminal connected by a communications link to a timesharing or other large computer system. In such an environment, the software advantages of the large system and the hardware advantages of the small system can complement one another, and can — potentially at least — gain the advantages of both.

But, enough of the past and the future. What can GCS offer today to the typical user — e.g., a routine FORTRAN programmer at an installation which has a limited amount of money to invest in graphics?

The USMA Graphics Compatibility System (GCS) is a FORTRAN-based computer graphics system designed for interactive use on a wide variety of computer graphics terminals. Due to its comprehensive and modular design, GCS provides a simplified easy-to-learn and easy-to-use approach to computer graphics, while simultaneously provid-

ing a powerfull tool which the sophisticated programmer may use for demanding and highly interactive graphical applications.

GCS provides compatibility at two distinct levels:

A. Cross-compatibility from one computer systems to another; and,

B. Cross-compatibility between computer graphics terminals.

Cross-compatibility from one computer to another has been achieved by writing all GCS software in FORTRAN, with emphasis placed on strict adherence to the specifications established by the American National Standards Institute (ANSI).

Cross-compatibility from one terminal to another has been achieved in a manner which is completely transparent to the user. One need no longer be concerned about the problems of "tailoring" his programs for a given graphical device — GCS provides the maximum capabilities offered by the various terminals in order to satisfy the user's graphical requirements. Device independence provides the user with the option of preparing and debugging his graphics program on a terminal other than the type he may desire for final output — a consideration of paramount importance for installations which may have a limited number of graphical devices available at any given time.

It should be emphasized that GCS is not just another collection of unrelated graphics subroutines — it is a unified system which provides the user with the flexibility to perform his tasks at any level of involvement that he desires. Because of this unique man/software relationship, GCS can also be thought of as "user compatible," and it is in this last facet of compatibility that GCS unveils its true potential. Individuals who would normally dismiss graphics for their particular applications are now provided with an alternative. High-level, composite routines are provided to perform relatively complex but frequently required tasks such as the preparation of complete graphs and histograms. Flexibility and adaptability to the needs of the more sophisticated user are achieved by allowing such a user to control the various GCS options which are present to standard default conditions for the unsophisticated programmer. Users of all disciplines may feed equally comfortable with GCS, since they are required to interact only at their particular level of graphics involvement.

The GCS development was a response to a specific need. In the winter of 1971-72, it was decided that the U.S. Military Academy would teach a series of courses to senior R&D managers dealing with computer assisted design-engineering, with special emphasis upon interactive computer graphics.

It was decided that the course would be heavily, hands-on oriented and that USMA would demonstrate practical applications of graphics from each Math-Science-Engineering academic department area, programmed by a faculty member from that area. It was also decided that these applications should be spread over at least four types of graphics output:

1. Printer-plot graphics — both in batch mode and from a timeshare terminal.

2. Pen-plot graphics — both interactive and non-interactive.

3. Static or add-on CRT graphics — storage tube display.

4. Dynamic CRT graphics — refresh tube display.

The Army Material Command (AMC) was to supply adjunctive graphics equipment to add to the existing USMA Academic Computer Center timesharing systems: USMA was to provide the teaching faculty and develop both the educational software and the required

computer software; the U.S. Army Computer Systems Command was to provide a Liaison Officer who would assist in the computer software development.

The course was to begin in July 1972. The new graphics equipment would not be available until April or May. Terminal equipment would arrive so late that programs to run on one type of graphics terminal might have to be checked out by testing it on another type of terminal.

Tha academic department faculty members would not be available to work on the project until the start of June. Most of them were reasonably competent FORTRAN programmers with little, if any, prior experience with graphics — but most of them would have time to take a 6-8 lesson course in graphics during their lunch periods once a week in the spring semester.

In order to meet this near-impossible set of constraints, a system was devised which could be both quickly and easily learned by faculty members with varied levels of FORTRAN experience (and, in many cases, no FORTRAN experience) and could be built and checked out in a tremendous hurry, using no more than one or two full-time programmers, with part-time assistance of half a dozen or so other programmers.

GCS (Version 1.0) was the response. Somehow it was developed in time. It sputtered, wheezed, ran slowly, did not have very many sophisticated options, and had embedded in it various pieces of proprietary software which were not releasable to others. It was well accepted, however, by its users. It met the seminar requirements and was widely used during the Academic Year 1972-73, with a great deal of useful feedback being obtained. A year later, in the late spring of 1973, GCS version 2.0, rebuilt from the bottom up, became operational. It still ran the same program, with occasional minor changes suggested by users as human engineering improvements. It was faster, supported more advanced options, was much more solid, and no longer contained even the most remote trace of any proprietary software in its modules.

In October 1973, GCS had progressed to the stage that it was ready to release for field testng by other agencies. As a result of this field testing, some interesting and valuable suggestions were incorporated into GCS. In January 1975, the U.S. Military Academy was no longer able to support GCS and the responsibility for maintenance, distribution, and future development was transferred to the U.S. Army Corps of Engineers Waterways Experiment Station.

CHAPTER I

IMPORTANT TECHNICAL CONCEPTS AND CONVENTIONS

In order to understand and use a system as powerful and versatile as GCS, the user should be aware of certain basic technical concepts and conventions.

Relationship of GCS to FORTRAN

The Graphics Compatibility System (GCS) is a package of inter-related subroutines written in ANSI FORTRAN. In order to produce graphics output, a FORTRAN program is written to perform whatever computations and/or graphics manipulations are needed to produce the desired image. Embedded in the program are calls to GCS subroutines to handle the graphic display and interaction, if relevant.

Under normal conditions, all input-output between the computer and graphics devices such as plotters and terminals will be handled via calls to GCS routines, without recourse to system dependent routines.

The programming conventions of GCS are identical to those of ANSI FORTRAN except that the use of GCS creates additional reserved names. All GCS subroutines designed for *User* use are FORTRAN subroutines beginning with the letter *U*. The GCS system also includes additional internal subroutines not intended for availability to users which always begin with the letters *GCS*. Thus the user, to avoid potential confusion and/or conflict between his program and GCS, should avoid creating his own subroutines with names beginning with either U or GCS.

All numbers used as calling parameters are passed to GCS as real numbers, even in situations where they represent integers.

Graphics Status Area (GSA) - Preset Modes and Parameters

In order to keep continuous track of many items of information relevant to graphics activities such as where the beam (or pen) is currently located, what portion of the screen (or plotting area) is currently within limits and what portion is currently off limits for drawing, what portion is within limits and what portion is off limits for textual material, etc, a labelled COMMON area GSA is established. The GSA common block also keeps track of the user's current choice from among the list of available options which define or mode of operation of the system.

When one starts a GCS program, the user's start routine *(USTART)* automatically sets all options to a standard initial condition: rectangular coordinate system, absolute (rather than relative or incremental), plotting angles to be measured in degrees, lines to be drawn routinely as solid lines without special attributes, etc.

The following is a sample program which illustrates the use of the GCS package to produce graphics output.

```
      DIMENSION X(2),Y(2)

      A = 75.
      B = 75.
      X(1) = 25.
      Y(1) = 75.
      X(2) = 75.
      Y(2) = 25.                    COMMENTS
```

```
CALL USTART                  'A'
CALL UOUTLN                  'B'
CALL UMOVE (25.,25.)         'C'
CALL UPEN (A,B)              'D'
CALL UMOVE (X(1),Y(1))       'E'
CALL USET ('DASH')           'F'
CALL UPEN (X(2),Y(2))
CALL UEND                    'G'
STOP
END
```

COMMENTS:

A: The first GCS call must be to USTART, a subroutine to initialize the status of the system.

B: UOUTLN will be explained in Chapter III.

C: All user oriented GCS subroutines begin with the letter 'U' and have easy to remember names.

D: Simple pen movements, such as visible UPEN's and invisible UMOVE's, are dependent upon the status of the system for their resulting actions.

E: All arguments to GCS subroutine calls are either **real** or **character**.

F: The status of the system can be modified, as in this case with USET, in order to obtain a wide range of output types.

G: The last GCS call must be to UEND, a subroutine which performs any termination activity.

GCS CONVENTIONS

Easy to remember English-language descriptions rather than arbitrary codes are used to specify modes. For example, if one is in the preset condition and wishes to draw in polar coordinates with angles measured in radians and lines drawn as vectors (solid lines with arrowheads), one would merely use the following *U*ser mode-*Set*ting commands:

```
CALL USET ('POLAR')
CALL USET ('RADIANS')
CALL USET ('VECTOR')
```

To revert to drawing in rectangular coordinates with plain lines, the commands would be:

```
CALL USET ('RECTANGULAR')
CALL USET ('LINE')
```

Furthermore, there are those modes which require an additional parameter to be associated with that particular mode. For example, the preset polynomial degree to which a set of data can be fitted (using the subroutine ULSTSQ) is 5. If the user would prefer that his data be fitted to a polynomial of degree 3, he would use the *U*ser *P*arameter *Set*ting command:

```
CALL UPSET ('POLYNOMIAL DEGREE',3.)
```

(before the call to ULSTSQ). To re-establish the original polynomial of degree 5, the appropriate call would be:

CALL UPSET ('POLYNOMIAL DEGREE',5.)

The user need never be concerned with these modes of parameters unless the standard values do not suit him. He can easily change them to values he wants. Then he can forget all about them until he once again feels a need to change, or until the GSA is reinitialized.

The underlying principle which permits the use of this option setting technique is the fact that the GSA is in fact a table containing all of the options necessary to define an environment in which graphics activities are performed. These elements are preset to values which reflect the most commonly performed graphics operations. In order to make a particular graphics option, only one element of the table is altered.

It will be normal practice in this manual to spell out completely the USET or UPSET option-names in full even if the name is very long, such as 'DOUBLEARROW' or 'DASHEDLINE'. Doing this is good form and good self-documentation for programs. GCS will, however, analyze only the first 4 characters in any option-name designator. If misspellings, truncations or non-standard wording occurs, the corrupted version will be fully acceptable to GCS provided that the first 4 characters of the character string are accurate.

With this system of organization, the beginning user—or the infrequent user—is insulated from the adverse effects which would tend to overwhelm the beginner in a system designed to provide flexibility for advanced and sophisticated users.

The user need be aware of, and pay attention to, only those degrees of freedom which he specifically wants to exploit in writing his program. GCS does not burden the user with long, complicated calling-parameter lists and complicated arbitrary codes. It can use short, easy-to-remember and easy-to-use commands convenient to both unsophisticated and sophisticated users.

Information in This Manual and Information Excluded From It

Some idea of the scope and range of the subroutines and options most likely to be of interest to the beginning user may be found in Appendices 'A' and 'B'. Sophisticated subroutines and options such as those involved in dealing with axis transformations, storing, manipulating and editing pictorial data structures have been completely omitted from this manual. Only a very limited subset of the most important and most frequently used subroutines and options from these appendices will actually be described in detail and discussed in this manual. For more complete programming information, see the separate 'GCS Programmer Reference Manual'.

CHAPTER II

GCS PROGRAMMING FUNDAMENTALS

Initialization

The first GCS statement in any program must be

CALL USTART

Its functions are many, varied and important. Some of them are:

A. It prepares the graphic terminal for plotting. In doing so, it clears the screen (or, in the case of plotter, requests the attendant to place a fresh sheet of paper on the plotbed), places the beam (or pan) in a standard position—coordinates (0,0) at the lower left corner of the standard plotting area.

B. It sets all mode and parameter options at their standard default values.

In the standard default condition, the system is set to use Cartesian, absolute coordinates and to draw solid lines in a large square area contained within the screen or plotting bed of the graphics device. It is preset to accept the values of $0. < X < 100.$ and $0. < Y < 100.$ Additional default settings by USTART will be mentioned later.

Reinitialization

At any time the initial default conditions can be restored by:

CALL URESET

Termination

It is necessary to perform various file and buffer termination activities upon exit from GCS by putting as the last GCS command of any program

CALL UEND

Use of this statement will insure that all GCS output is completed.

Erasing or Starting Off Again With a Clean Sheet

At various stages in running a program, it is often worthwhile to erase everything which has been drawn or printed on a CRT screen or to ask the attendant of a plotter to replace the old sheet on the plotter with a fresh clean one. The call for performance of this function is:

CALL UERASE

The pen position remains unchanged when erasure occurs.

Interactive Versus Batch

In a batch program there is no need to suspend execution in order to view multiple plots

prior to erasing or starting off again with a clean sheet. In an interactive program an alarm can be sounded to indicate a plot is done by:

CALL UBELL

Then to allow viewing of the plot use:

CALL UPAUSE

Simple Line-Drawing

The basic command in GCS for drawing lines is

CALL UPEN (X,Y)

This routine moves the beam (or pen) from its present coordinate position **(X0,Y0)** to a designated new location (X,Y). Pen status variables in the Graphics Status Area (GSA) determine what kind of line (if any) is drawn as a result of this movement.

Until some sort of overt action is taken to change pen status, every UPEN movement will cause a solid line to be drawn. (CALL USTART sets pen status to 'LINE'.) CALL USTART also sets the initial beam (pen) position to (0,0). Example II-1 shows a simple series of CALL UPEN commands which will draw a square by 4 basic pen-movements.

Invisible Lines

If one wanted to draw a similar square starting at some location other than the origin, one needs to be able to move the beam or pen invisibly, that is, without drawing any line. One way to do this is to use the UMOVE subroutine

CALL UMOVE (X,Y)

The effect is identical to UPEN except that no line of any kind is drawn as the beam or pen moves from the old location **(X0,Y0)** to the new location (X,Y). Example II-2 shows a program which draws a square offset from the origin in this way. In this example we have used subroutine UOUTLN to outline the square we are drawing within. (UOUTLN is further explained in Chapter III.) The default origin is at the lower left corner (0,0).

Using a Pen Status Mode Change to Draw Invisible Lines

The same effect as a call to UMOVE can be achieved by changing pen status mode from its default condition to 'NOLINE' by means of a call to USET.

CALL USET ('NOLINE')

causes the system to again draw a line. Example II-3 shows a program which uses this approach to draw the same image as Example II-2.

Another subroutine that is considered to be a simple line drawing subroutine, capable, if desired, of producing an invisible line as in the example above. A call to

CALL UPEN1 (X,Y,'NOLINE')

enables the user to generate an invisible line to (X,Y). The mode will be established as indicated for only that 1 pen movement. Effectively, the above call is the same as the CALL UMOVE (X,Y), or of the CALL USET ('NOLINE'), CALL UPEN (X,Y), CALL USET

('LINE') sequence. It will be shown throughout this text, however, that the chosen mode can be one of many different line modes other than 'NOLINE'.

Pen Status Mode Changes Allow Many Kinds of Lines to be Drawn

The UPEN subroutines of GCS can draw many other kinds of lines, including lines in various colors (if the plotting device permits), dashed lines, lines with tics along their length, and lines of all these types with various kinds of terminating characters or symbols. The number of permutations of line types and terminators to be used at the end of a line is very large. Some of the more important are demonstrated later in Chapter VI. However some of the basic options are considered below.

Alternate Colors

If the graphical device being used has the ability to display lines of different colors, the user can request that any subsequent line be generated with one of seven colors, using CALL USET (OPTION), where OPTION in this case is 'WHITE,' 'BLACK,' 'RED,' 'GREEN,' 'YELLOW,' 'BLUE,' 'MAGENTA,' or 'CYAN.'

Many terminals such as the Tektronix do not have a multi-color capability and, as such use the default color BLACK. Others, including most pen plotters, allow manual color change by replacement of the drawing pen by one of another color. A color mode change with such a device causes the plotter to position itself for pen replacement and a color change request to be typed out on the control device.

Most things done with colors can be done on devices which do not have a color capability by making distinctions through the use of plain lines, and various types of dashed lines, arrow lines, ticmarked lines and so forth.

Vectors or Lines with Arrowheads

One frequently useful set of options is the arrowhead options. These are useful for drawing vectors, dimension lines, etc. These use the following pen status modes:

Pen Status	Description
'ARROW'	Draws a "forward vector" with its arrowhead at (X,Y)
'BACKARROW'	Draws "backward vector" with its arrowhead at (Xo,Yo)
'DOUBLEARROW'	Draws a "dimension line," i.e., a line with arrowhead pointing out at each end.

Example II-4 demonstrates the use of the 'ARROW' option.

Tic Marked Lines

Another available option is tic lines, i.e., lines with tic marks at specified intervals along it. Pen status may be set to draw such lines by using a call to USET.

```
CALL USET ('TICLINE')
```

A call to UPSET

```
CALL UPSET ('TICINTERVAL',PARAMETER)
```

will change the interval along a ticline at which the tics will actually be placed. For example:

```
CALL UPSET ('TICINTERVAL',5.)
CALL UPEN1 (X,Y,'TICLINE')
```

will produce a ticline to (X,Y), marked off with tics at intervals of 5 units rather than the default value of 10. Example II-5 shows several ticline options.

Dashed Lines

One can draw dashed lines by going to the 'DASHLINE' mode with a

```
CALL USET ('DASHLINE')
```

The length of the dashes and or the intervals between the dashes can be modified, if desired, with a

```
CALL UPSET ('SETDASH', DASHCODE)
```

the resulting dashed line will be repetitions of the effect caused by the combination of the digits (0-9) as follows:

DASHCODE Digit	Effect on Visible Units	Effect on Invisible Units
1	1	—
2	—	1
3	2	—
4	—	2
5	5	—
6	—	5
7	10	—
8	—	10
9	1 point	—
0	—	1 point

The specification of a single digit for DASHCODE will produce standard dashed (DASHCODE = 1-8) or standard dotted (DASHCODE = 9) lines. If the user desires a non-standard dashed line he must specify a DASHCODE of two or more digits, followed by a decimal point. The following sequence:

```
CALL USET ('DASH')
CALL UPSET ('SETDASH',76.)
CALL UPEN (X,Y)
```

will produce for example, a dashed line to (X,Y) consisting of 10 visible units, 5 invisible units, 10 visible units, 5 invisible units, etc. The physical length of a unit will be approximately 0.04 inches for the majority of the devices. Example II-6 illustrates several dashed line options. The number of possible types of dashed lines is limited only by the word length of the central computer.

Polar Coordinates

The options absolute and rectangular (Cartesian) are established as defaults by USTART, because most drawings use this simple coordinate system. A polar coordinate system provides the user with the capability of referring to a location on the device in polar coordinates (R,THETA) rather than in rectangular coordinates (X,Y). Example II-7 illustrates use of polar plotting.

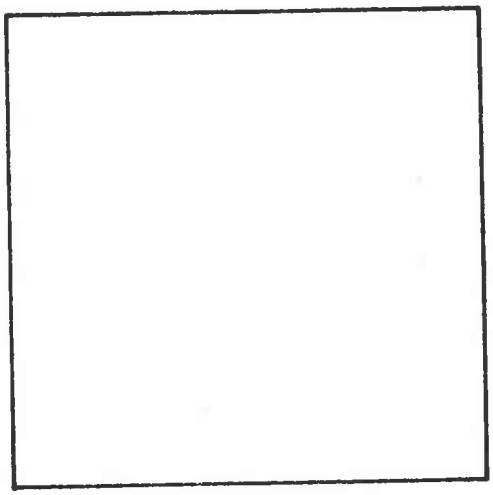
Relative (Incremental) Plotting

The relative coordinate mode interprets the coordinate pair of the pen request as units relative (positive or negative) to the previous pen position rather than as an absolute position. Example II-8 illustrates use of relative plotting.

```
C     THIS SAMPLE PROGRAM WILL DEMONSTRATE SIMPLE LINE-DRAWING BY
C     DRAWING A SQUARE BOX IN 4 PEN MOVEMENTS
C
C     INITIALIZE GCS
C     THIS INITIALIZATION SETS GCS TO RECTANGULAR, ABSOLUTE
C     COORDINATES, SOLID LINE PEN-DRAWING MODE, AND INITIAL PEN
C     COORDINATES (0,0)
C
      CALL USTART
C
C     NOTE THAT ALL GCS SUBROUTINES (INCLUDING THE UPEN ROUTINE USE
C     TYPE REAL CALLING PARAMETERS.  THUS COORDINATES MUST BE
C     ENTERED AS REAL NUMBERS, I.E. WITH DECIMAL POINTS
C
C     MOVE PEN TO (0,50) THEREBY DRAWING LINE (0,0) to (0,50)

      CALL UPEN (0.,50.)
C
C     MOVE PEN TO (50,50) THEREBY DRAWING LINE (50,50) TO (50,0)
C
      CALL UPEN (50.,0.)
C
      MOVE PEN TO (0,0) THEREBY DRAWING LINE (50,0) TO (0,0)
C
      CALL UPEN (0.,0.)
C
C     THIS COMPLETES DRAWING OF THE SQUARE
C
C     WRAP-UP.  FIRST TERMINATE GCS BY CALL UEND.  THEN STOP EXECUTION
C     WITH STOP.  FINALLY END FORTRAN PROGRAM WITH END.
C
      CALL UEND
      STOP
      END
```

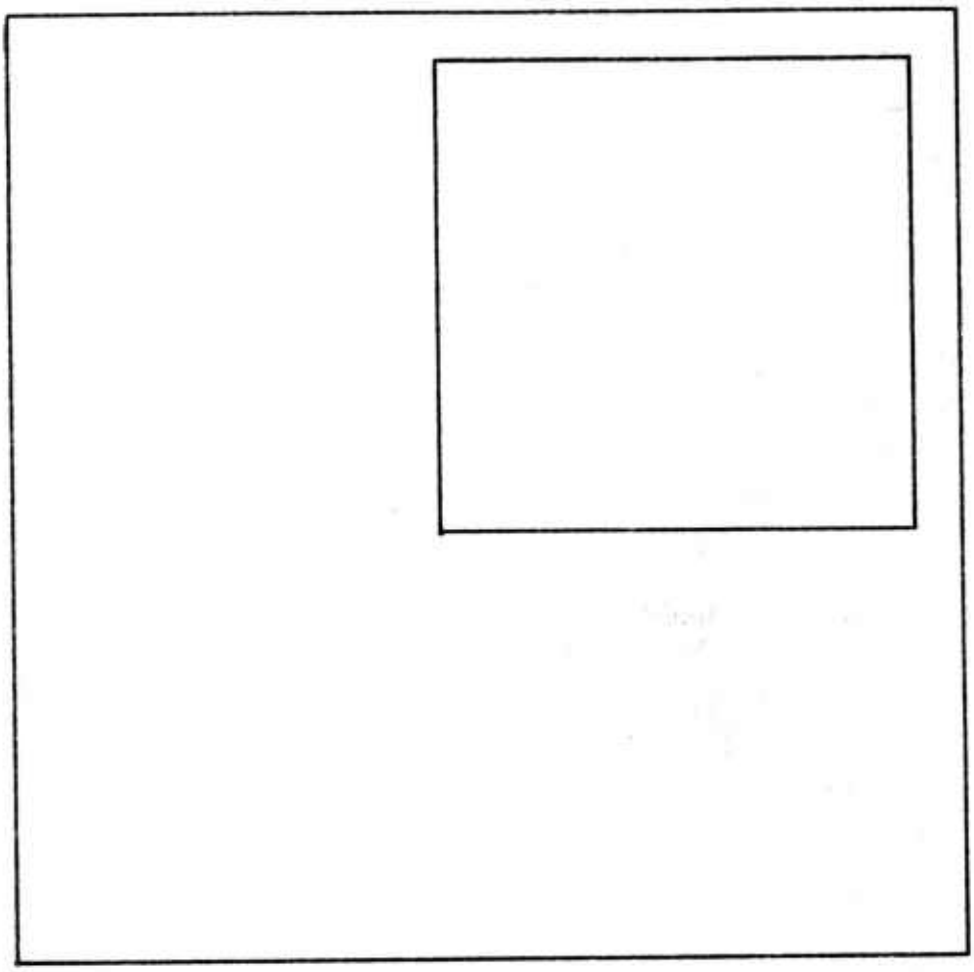
EXAMPLE II-1



```
CALL USTART
CALL UPEN (0.,50.)
CALL UPEN (50.,50.)
CALL UPEN (50.,0.)
CALL UPEN (0.,0.)
CALL UEND
STOP
END
```

```
C     THIS PROGRAM DEMONSTRATES USE OF THE MOVE COMMAND TO MOVE
C     THE PEN INVISIBLY WITHOUT NEED FOR A MODE CHANGE.  IT DRAWS A
C     SQUARE IDENTICAL TO THE PREVIOUS ONE.
C
C     INITIALIZE
C
      CALL USTART
      CALL UOUTLN
C
C     MOVE PEN INVISIBLY TO COORDINATES (45,45)
C
      CALL UMOVE (45.,45.)
C
C     NO CHANGE HAS BEEN MADE IN PENSTATUS SO IT IS STILL IN THE DEFAULT
C     CASE OF SOLID LINES. DRAW THE SQUARE.
C
      CALL UPEN (45.,95.)
      CALL UPEN (95.,95.)
      CALL UPEN (95.,45.)
      CALL UPEN (45.,45.)
C
C     WRAP UP
C
      CALL UEND
      STOP
      END
```

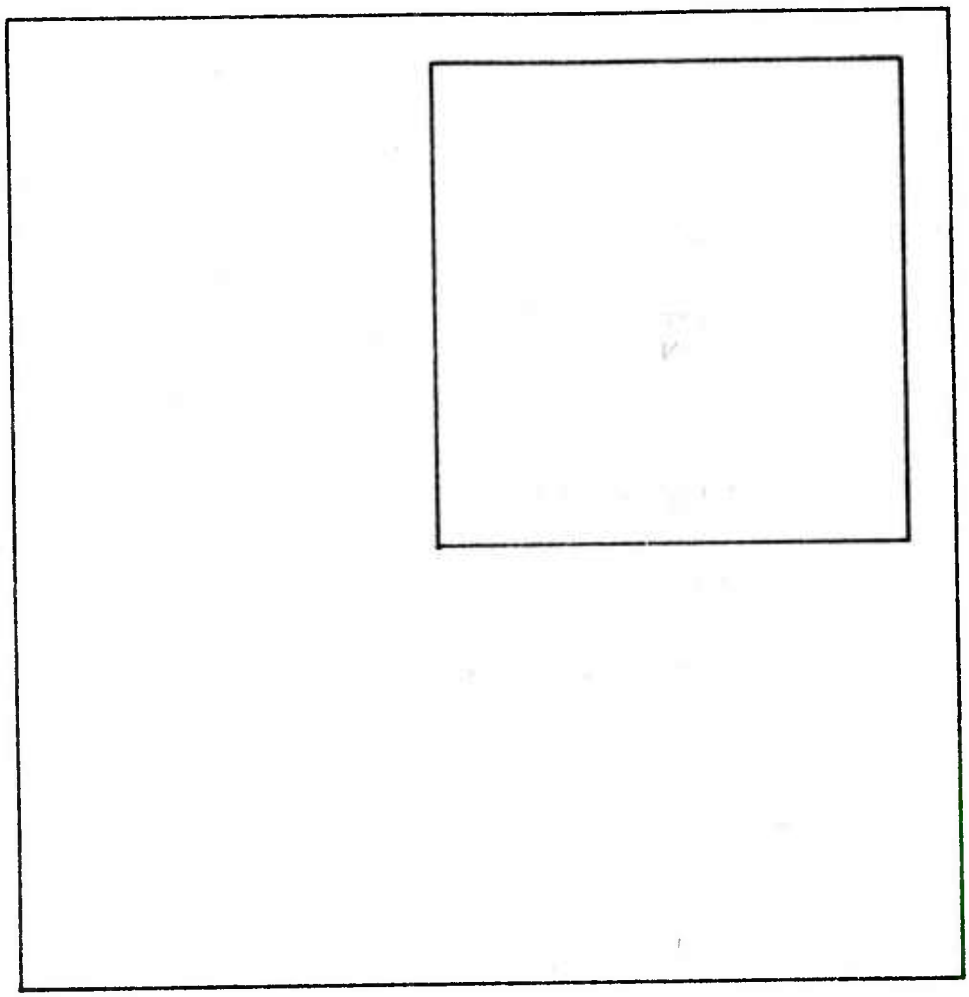
EXAMPLE II-2



```
CALL USTART
CALL UOUTLN
CALL UMOVE (45.,45.)
CALL UPEN (45.,85.)
CALL UPEN (85.,85.)
CALL UPEN (85.,45.)
CALL UPEN (45.,45.)
CALL UEND
STOP
END
```

```
C     THIS PROGRAM DEMONSTRATES THE USE OF A MODE CHANGE TO MOVE
C     PEN POSITION WITHOUT DRAWING A LINE.  OTHERWISE IT DRAWS A
C     SQUARE IDENTICAL TO PREVIOUS ONE.
C
C     FOLLOWING INITIALIZATION BY USTART IS ALWAYS NECESSARY.  AMONG
C     OTHER THINGS IT AUTOMATICALLY SETS PENSTATUS FOR DRAWING SOLID
C     LINES AND INITAL PEN POSITION TO COORDINATES (0,0).
C
      CALL USTART
      CALL UOUTLN
C
C     SET MODE TO 'NOLINE' THEN MOVE PEN TO COORDINATES (45,45) WITHOUT
C     DRAWING A LINE
C
      CALL USET ('NOLINE')
      CALL UPEN (45.,45.)
C
C     NOW RESET PENSTATUS FOR DRAWING SOLID LINES AND DRAW SQUARE
C
      CALL USET ('LINE')
      CALL UPEN (45.,95.)
      CALL UPEN (95.,95.)
      CALL UPEN (95.,45.)
      CALL UPEN (45.,45.)
C
C     WRAP UP
C
      CALL UEND
      STOP
      END
```

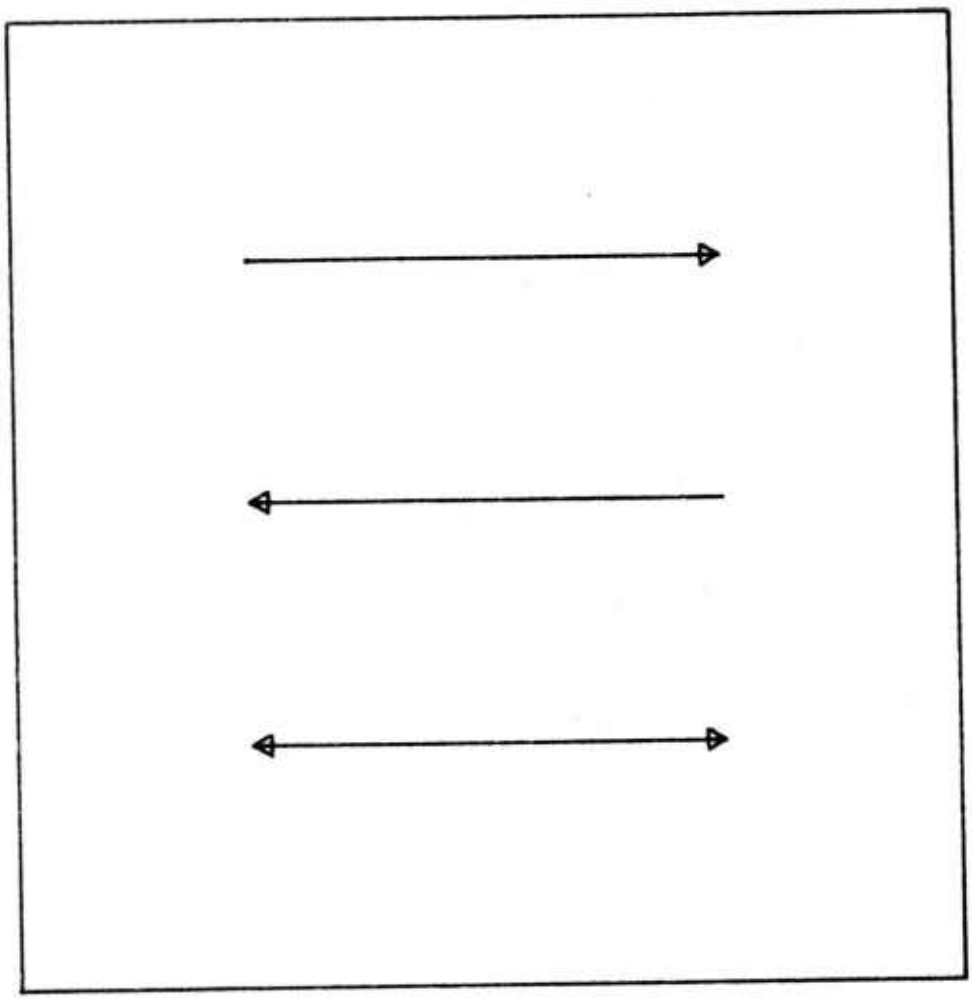
EXAMPLE II-3



```
CALL USTART
CALL UOUTLN
CALL USET ('NOLINE')
CALL UPEN (45.,45.)
CALL USET ('LINE')
CALL UPEN (45.,95.)
CALL UPEN (95.,95.)
CALL UPEN (95.,45.)
CALL UPEN (45.,45.)
CALL UEND
STOP
END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE 'ARROW', 'BACKARROW', AND
C     'DOUBLEARROW' LINE OPTIONS AVAILABLE THROUGH USET/UPEN.
C     INITIALIZE GCS, AND THEN DRAW THE OUTLINE OF OUR PLOTTING AREA
C
      CALL USTART
      CALL UOUTLN
C
C     MOVE TO VIRTUAL LOCATION (25,75), SET LINE TYPE TO 'ARROW', AND
C     DRAW A LINE WHICH EXTENDS TO VIRTUAL LOCATION (75,75).
C
      CALL UMOVE
      CALL USET ('BACKARROW')
      CALL UPEN (75.,50.)
C
C     MOVE BEAM/PEN TO VIRTUAL LOCATION (25,25), SET LINE TYPE TO
C     'DOUBLEARROW', AND THEN DRAW LINE WHICH EXTENDS TO VIRTUAL
C     LOCATION (75,25).  TERMINATE THE PROGRAM AFTER LINE IS DRAWN.
C
      CALL UMOVE (25.,25.)
      CALL USET ('DOUBLEARROW')
      CALL UPEN (75.,25.)
      CALL UEND
      STOP
      END
```

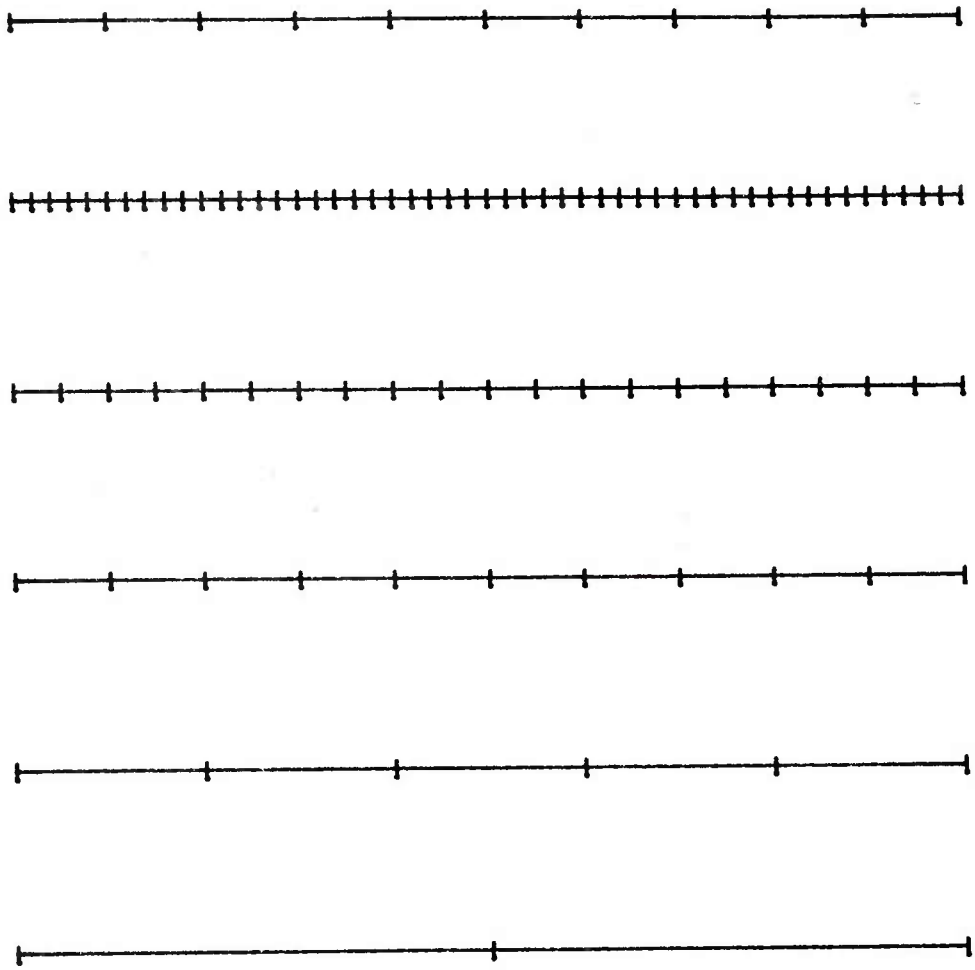
EXAMPLE II-4



```
CALL USTART
CALL UOUTLN
CALL UMOVE (25.,75.)
CALL USET ('ARROW')
CALL UPEN (75.,75.)
CALL UMOVE (25.,50.)
CALL USET ('BACKARROW')
CALL UPEN (75.,50.)
CALL UMOVE (25.,25.)
CALL USET ('DOUBLEARROW')
CALL UPEN (75.,25.)
CALL UEND
STOP
END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE TIC LINE GENERATION PTIONS
C     AVAILABLE THROUGH GCS.  INITIALIZE GCS, SET PEN-STATUS TO THE 'TIC'
C     MODE, AND THEN DRAW A LINE WHICH BEGINS AT (0.,99.) AND TERMINATES
C     AT (100.,99.) USING THE DEFAULT TIC LENGTH.
C
      CALL USTART
      CALL USET ('TICLINE')
      CALL UMOVE (0.,99.)
      CALL UPEN (100.,99.)
C
C     REQUEST TICS TO APPEAR AT EVERY 5.0 VIRTUAL UNITS, AND DRAW A LINE
C     WHICH STARTS AT (0.,60.) AND ENDS AT (100.,60.).
C
      CALL UPSET ('TICINTERVAL',5.)
      CALL UMOVE (0.,60.)
      CALL UPEN (100.,60.)
C
C     REQUEST TICS TO APPEAR AT EVERY 10. VIRTUAL UNITS, AND DRAW A LINE
C     WHICH STARTS AT (0.,40.) AND ENDS AT (100.,40.).
C
      CALL UPSET ('TICINTERVAL',10.)
      CALL UMOVE (0.,40.)
      CALL UPEN (100,.20)
C
C     REQUEST TICS TO APPEAR AT EVERY 20.  VIRTUAL UNITS, AND DRAW A LINE
C     WHICH STARTS AT (0.,20.) AND ENDS AT (100.,20.).
C
      CALL UPSET ('TICINTERVAL',20.)
      CALL UMOVE (0.,20.)
      CALL UPEN (100.,20.)
C
C     REQUEST TICS TO APPEAR AT EVERY 50. VIRTUAL UNITS, AND DRAW A LINE
C     WHICH STARTS AT (0.,1.) AND ENDS AT (100.,1.).
C
      CALL UPSET ('TICINTERVAL',50.)
      CALL UMOVE (0.,1.)
      CALL UPEN (100.,1.)
      CALL UEND
      STOP
      END
```

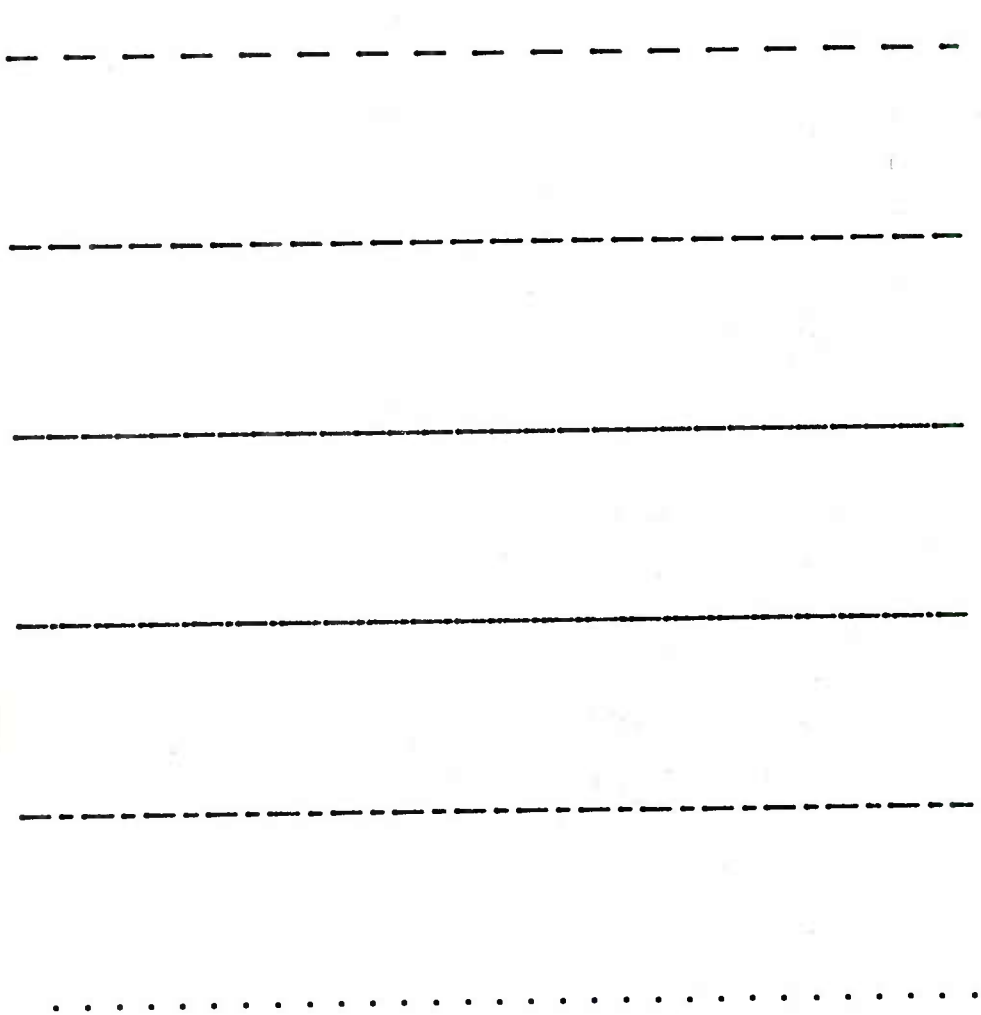
EXAMPLE II-5



```
CALL USTART
CALL USET ('TICLINE')
CALL UMOVE(0.,99.)
CALL UPEN(100.,99.)
CALL UPSET ('TICINTERVAL',2.)
CALL UMOVE (0.,80.)
CALL UPEN (100.,80.)
CALL UPSET ('TICINTERVAL',5.)
CALL UMOVE (0.,60.)
CALL UPEN (100.,60.)
CALL UPSET ('TICINTERVAL',10.)
CALL UMOVE (0.,40.)
CALL UPEN (100.,40.)
CALL UPSET ('TICINTERVAL',20.)
CALL UMOVE (0.,20.)
CALL UPEN (100.,20.)
CALL UPSET ('TICINTERVAL',50.)
CALL UMOVE (0.,1.)
CALL UPEN (100.,1.)
CALL UEND
STOP
END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE DASHED LINE GENERATION
C     OPTIONS AVAILABLE THROUGH GCS. INITIALIZE GCS, SET THE PEN-STATUS
C     TO 'DASH' MODE, AND THEN DRAW A LINE WHICH BEGINS AT (0.,100.) AND
C     TERMINATES AT (100.,100.). THE DEFAULT VALUE OF DASH WILL BE USED
C     FOR THIS CASE.
C
      CALL USTART
      CALL USET ('DASHLINE')
      CALL UMOVE (0.,100.)
      CALL UPEN (100.,100.)
C
C     SET THE DASH SPECIFICATION TO 54., AND DRAW A LINE THAT STARTS AT
C     (0.,80.) AND ENDS AT (100.,80.).
C
      CALL UPSET ('SETDASH',54.)
      CALL UMOVE (0.,60.)
      CALL UPEN (100.,60.)
C
C     SET THE DASH SPECIFICATION TO 52., AND DRAW A LINE THAT STARTS AT
C     (0.,60.) AND ENDS AT (100.,60.).
C
      CALL UPSET ('SETDASH',52.)
C
C     SET THE DASH SPECIFICATION TO 5212., THEN GENERATE A LINE THAT
C     STARTS AT (0.,40.) AND ENDS AT (100.,40.).
C
      CALL UPSET ('SETDASH',5212.)
      CALL UMOVE (0.,40.)
      CALL UPEN (100.,40.)
C
C     SET THE DASH SPECIFICATION TO 5434., THEN GENERATE A LINE THAT
C     STARTS AT (0.,20.) AND ENDS AT (100.,20.).
C
      CALL UPSET ('SETDASH',5434.)
      CALL UMOVE (0.,20.)
      CALL UPEN (100.,20.)
C
C     SET THE DASH SPECIFICATION TO 96., AND DRAW A LINE THAT STARTS AT
C     (0.,0.) AND ENDS AT (100.,0.).
C
      CALL UPSET ('SETDASH',96.)
      CALL UMOVE (0.,0.)
      CALL UPEN (100.,0.)
      CALL UEND
      STOP
      END
```

EXAMPLE II-6



```
CALL USTART
CALL USET ('DASHLINE')
CALL UMOVE (0.,100.)
CALL UPEN (100.,100.)
CALL UPSET ('SETDASH',54.)
CALL UMOVE (0.,80.)
CALL UPEN (100.,80.)
CALL UPSET ('SETDASH',52.)
CALL UMOVE (0.,60.)
CALL UPEN (100.,60.)
CALL UPSET ('SETDASH',5212.)
CALL UMOVE (0.,40.)
CALL UPEN (100.,40.)
CALL UPSET ('SETDASH',5434.)
CALL UMOVE (0.,20.)
CALL UPEN (100.,20.)
CALL UPSET ('SETDASH',96.)
CALL UMOVE (0.,0.)
CALL UPEN (100.,0.)
CALL UEND
STOP
END
```

```
C       THIS PROGRAM WILL DEMONSTRATE THE USE OF POLAR PLOTTING BY
C       DRAWING A QUARTER OF A CIRCLE
C
C       INITIALIZE LINE DRAWING SPEED OF TERMINAL TO 120 CHARACTERS PER
C       SECOND
C
        CALL USTART
        CALL UPSET ('SPEED',120.)
        CALL UERASE
C
C       INDICATE PLOTTING WILL BE IN POLAR COORDINATES (R,θ)
C
        CALL USET ('POLAR')
C
C       PLOT A QUARTER OF A CIRCLE THAT HAS A RADIUS OF 100 IN NINE DEGREE
C       INCREMENTS FROM ZERO TO NINETY DEGREES
C
        RADIUS=100.
        DO 100 I=1,11
        ANGLE=FLOAT (I-1) * 9.
  100   CALL UPEN (RADIUS,ANGLE)
        CALL UPEN (0.,0.)
C
C       TERMINATION
C
        CALL UEND
        STOP
        END
```

EXAMPLE II-7

```
C     THIS PROGRAM WILL DEMONSTRATE THE USE OF RELATIVE PLOTTING
C
C     INITIALIZE LINE DRAWING SPEED OF TERMINAL TO 120 CHARACTERS PER
C     SECOND
C
      CALL USTART
      CALL UPSET ('SPEED',120.)
      CALL UERASE
C
C     INDICATE PLOTTING WILL BE DONE IN RELATIVE MODE AND DRAW A BOX
C     100 DATA UNITS ON A SIDE
C
      CALL USET ('RELATIVE')
      CALL UPEN (100.,0.)
      CALL UPEN (0.,100.)
      CALL UPEN (-100.,0.)
      CALL UPEN (0.,-100.)
C
C     TERMINATION
C
      CALL UEND
      STOP
      END
```

EXAMPLE II-8

CHAPTER III

VIRTUAL AND DEVICE GRAPHICS

Outlining the Plotting Area

To obtain an outline of the area within which all graphical activity will be performed, the programmer may invoke the GCS subroutine UOUTLN:

```
CALL UOUTLN
```

Although the use of UOUTLN certainly is not necessary for most graphical applications, there are many times (particularly during the learning and debugging phases) when it is useful for the programmer to know the physical boundaries of the display he is generating.

The Default Case and Simple Extensions

The examples presented so far have assumed the use of variables which fall into the range of values:

```
0 .LE. X .LE. 100

0 .LE. Y .LE. 100
```

with no specific control over where on the screen (or plotting bed) the pictures are actually drawn. This is the default case established by USTART.

The user can easily change the range of permissible values to the ranges

```
XMIN .LE. X .LE. XMAX

YMIN .LE. Y .LE. XMAX
```

by use of the command

```
CALL UWINDO (XMIN,XMAX,YMIN,YMAX)
```

This permits the user to plot anywhere in the space defined by the the four parameters in the subroutine call.

The values of the four parameters may be any FORTRAN real numbers permissible on the particular computer in use (subject only to the restriction XMIN .NE. XMAX and YMIN .NE. YMAX). See Example III-1.

Increasing and Decreasing Apparent Picture Size

Through the use of UWINDO, it is possible to increase or decrease the apparent picture size without having to change the values of the parameters required to generate the display. This facility effectively provides an elementary "zoom" capability which may be readily utilized under GCS. Example III-2 illustrates a simple six-step "zoom-out" by successively increasing the window specifications prior to generating each of the figures. Note that the pen commands required to draw the figure are identical in each of the six steps, and that the only dynamic entity is the window itself. Example III-3 illustr-

ates a six-step "zoom-in" through decrementing the virtual window. It should be noted that an *increase* in the window specifications *decreases* the apparent picture size (zoom-out) and that a *decrease* of the window specifications yields an *increase* in apparent picture size (zoom-in).

Distortion

It is important to note the ratio of the X-to-Y window specifications when defining new window boundaries, as ratios other than unity indicate that a distorted window will be contructed. Example III-4 illustrates this type of distortion by displaying the same basic information provided by Example III-3 with the exception that non-square windows are utilized.

Clipping the Edges of the Picture

Should the graphics programmer attempt to display information at coordinate locations which lie outside of the window boundaries, GCS will "clip" the display at the window boundaries and any graphical activity attempted outside of the boundaries will not be displayed. Example III-4 illustrates the clipping feature applied to a simple six-level "zoom-in". We will again return to the topic of clipping after we first consider the concept of device coordinate addressing under GCS.

Expanding Flexibility and Capabilities: Device Mode

When desired, GCS provides means of very detailed control of where pictures are drawn on the physical device. This may be done by going to the 'DEVICE' mode with a

CALL USET ('DEVICE')

Thereafter all coordinates are measured directly in device measurement units from the lower left corner of the display screen or plotting bed.

Several optional units of measure are available on the display screen or plotting bed:

CALL USET ('CENTIMETERS') causes horizontal and vertical distances to be measured in centimeters.

CALL USET ('FONTUNITS') causes horizontal measurement in units of the width of standard alphabetic characters (usually the hardware characters for the device being used) and vertical measurement in units of the height of the alphabetic characters.

CALL USET ('PERCENTUNITS') cause horizontal measurement in units of a percentage of the full width of the screen or plotting bed and vertical measurement in units of one percent of the full physical height of the screen or plotting bed.

CALL USET ('RASTERUNITS') causes horizontal and vertical measurement in terms of the smallest addressable unit for a particular device.

CALL USET ('INCHES') causes the horizontal and vertical measurement on the display to revert to the default case of inches.

See Example III-6

Viewporting: A More Flexible and Advanced Concept of Picture Control

Particular locations on the graphic device screen or plotting bed can be designated a device plotting area by the command

CALL UDAREA (XMIN,XMAX,YMIN,YMAX)

where XMIN,XMAX,YMIN and YMAX are measured in the currently defined units of measure in the device mode: inches, centimeters, percentunits, fontunits or rasterunits. Designation of this device plotting area has no effect in the device mode. However, in the virtual mode, either by default or by executing a call to USET ('VIRTUAL'), this device plotting area is at the heart of an extremely powerful concept of graphic control known as *viewporting*. This concept is most easily visualized by referring to the next figure while following the description below and the sample programs and their output.

The user may pick any scale or range of values desired to plot by means of a call to UWINDO. For example, to plot values in the range $0 < X < 1000$ and $0 < Y < 1000$, give the command

CALL UWINDO (0.,1000.,0.,1000.)

A call to UDAREA is used to specify where on the device to plot data. Then, plot in a 4-inch square at the left bottom of the display screen, first make sure that the unit of measure is 'INCHES' then give the command

CALL UDAREA (0.,4.,0.,4)

Next, make sure that the mode is 'VIRTUAL' and do any plotting desired. Any values within the "virtual window" 0 .LE. X,Y .LE. 1000 is projected into the corresponding position of the "device plotting area" 0 .LE. X,Y, .LE. 4. For example, the user's numeric values of Y = 0, X = 0 would map into the lower left corner of the screen or plotting bed at position X = 0, Y = 0. User values of X = 1000, Y = 1000 would map into the upper right corner of the 4-inch device plotting area at X = 4, Y = 4. User values of X = 250, Y = 750 would correspondingly map into X = 1, Y = 3.

But what happens if the user tries to draw a line to a point outside the 0 .LE. X, Y .LE. 1000 window, for example, X = 250, Y = 1500? Direct projection would indicate that such a line would rise vertically from the previous point to a point X = 1, Y = 6, but such a line would pass out of the designated device plotting area. What happens is that the line is 'clipped' off at the edge of the window, i.e., drawn only to the limit of the window (which is, of course, also the limit of the device plotting area). Thus if any other lines or textual material had previously been put into adjacent area just outside the designated plotting area it would be protected against being overwritten by data which was considered to be irrevelant because it fell outside the expressed values of plotting interest specified by UWINDO. This 'clipping' action associated with mapping from a virtual window to a specified device area adds markedly to the power and ease of pictorial control in GCS.

Here are some examples of the use of viewporting:

A. Convenient Units: the imaginary or expandable plotting area or virtual window can be defined to correspond to any convenient units which the user would like to think in terms of.

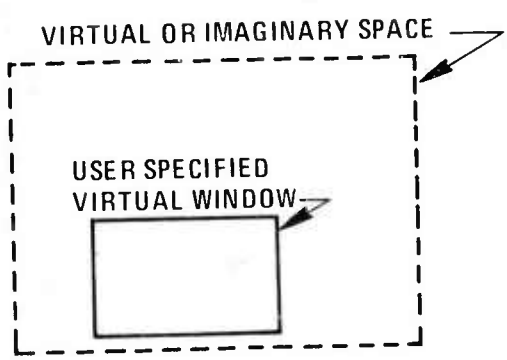
B. *Virtual Window Modification:* the user can selectively examine different areas of an entire display through the redefinition of the viewport followed by the pen movements of that entire display. This idea can also be used to examine a small area of a display in great detail.

C. *Device Window Modification:* the same display can be generated at many different locations on the device with the same pen movements by fixing the viewport and changing the device area, as illustrated in Example III-7. An extension of this device area modification capability allows the user to achieve distortions, as in Example III-8.

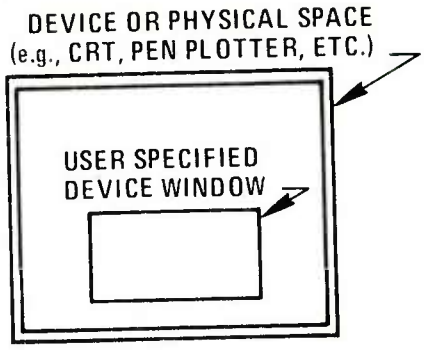
Device Independent Plotting

The previous examples that used subroutine UDAREA were written for a Tektronix 4010/4013 terminal. A device independent program can be written by using **'PERCENTUNITS'** or

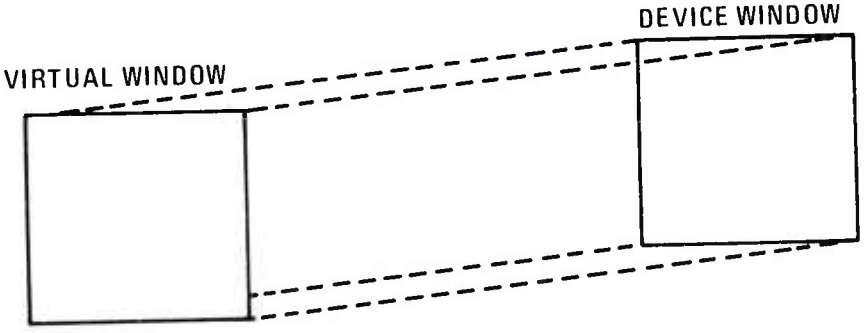
CALL USTUD (ARRAY)



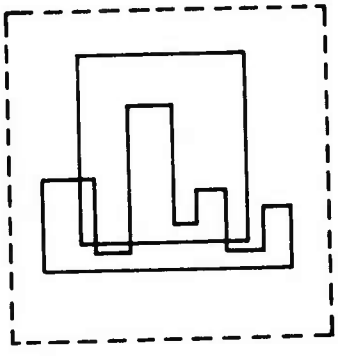
A USER'S CALL TO PEN MOVEMENTS ARE WITH RESPECT TO THE USER SPECIFIED VIRTUAL WINDOW.



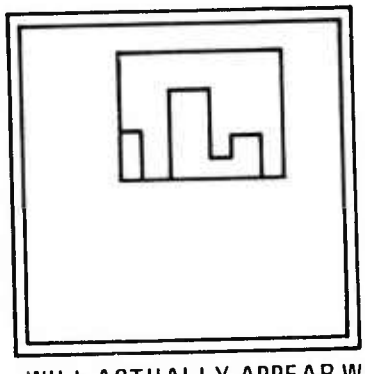
THESE PEN MOVEMENTS, IF WITHIN THE VIRTUAL WINDOW OF THE FIRST DIAGRAM, WILL ACTUALLY APPEAR WITHIN THE USER SPECIFIED DEVICE WINDOW OF THE ABOVE DIAGRAM.



THIS ACTUALLY OCCURS AS THE RESULT OF A MAPPING DIRECTLY FROM THE VIRTUAL WINDOW TO THE DEVICE WINDOW.



FOR EXAMPLE, THOSE PEN MOVEMENTS WHICH FALL WITHIN THIS PARTICULAR VIRTUAL WINDOW.....



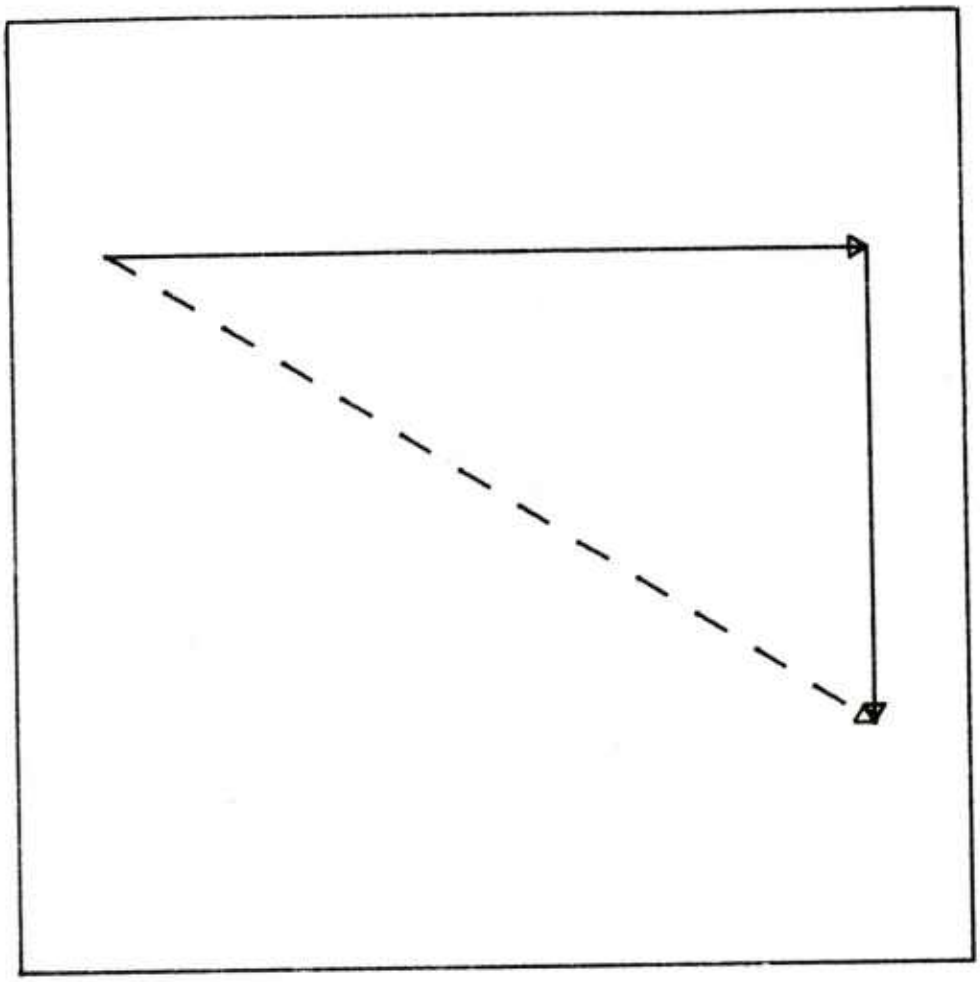
..... WILL ACTUALLY APPEAR WITHIN THIS PARTICULAR DEVICE WINDOW.

Figure 3-1. The Viewporting Concept

0791-1

```
C     THIS PROGRAM GENERATES TWO VECTORS WITH ARROW LINES, AND THE
C     RESULTANT VECTOR WITH A DASHED ARROW LINE.
C
C     INITIALIZE THE SYSTEM AND GENERATE AN OUTLINE
C
      CALL USTART
      CALL UOUTLN
C
C     REDEFINE THE VIRTUAL WINDOW
C     -50000 < X < 50000
C     0.00001 < Y < 0.000
C
      CALL UWINDO (-50000.,50000.,0.00001,0.00005)
C
C     DRAW THE TWO VECTORS
C
C     MOVE TO THE BEGINNING POINT OF THE FIRST VECTOR SET TO ARROW
C     MODE AND DRAW VECTOR FROM (-40000.,0.00004) TO (40000.,0.00004)
C
      CALL UMOVE (-40000.,0.00004)
      CALL USET ('ARROW')
      CALL UPEN (40000.,0.00004)
C
C     DRAW SECOND VECTOR FROM END OF FIRST TO (40000.,0.00002)
C
      CALL UPEN (40000.,0.00002)
C
C     MOVE TO BEGINNING OF VECTOR SYSTEM
C
      CALL UMOVE (-40000.,0.00004)
C
C     SET MODE TO DRAW DASHED ARROW AND DRAW RESULTANT VECTOR
C
      CALL USET ('DARROW')
      CALL UPEN (40000.,0.00002)
C
C     TERMINATION
C
      CALL UEND
      STOP
      END
```

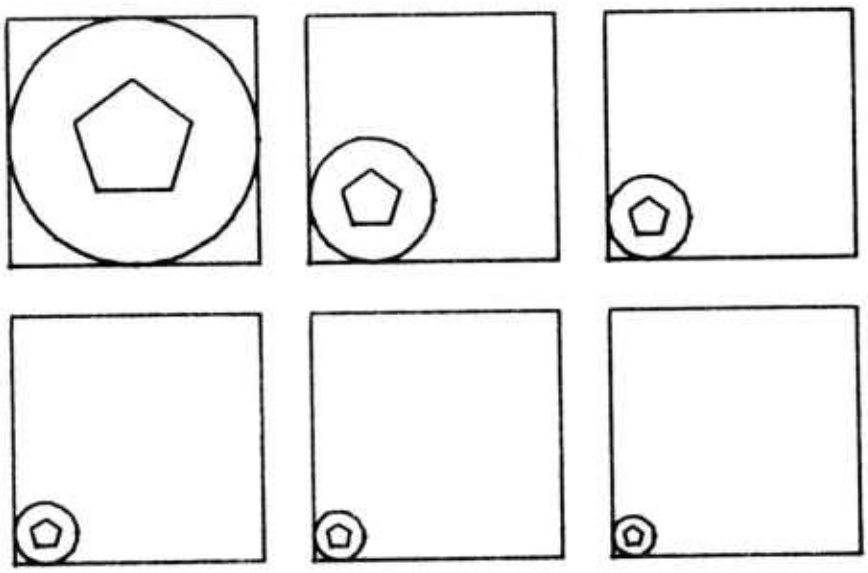
EXAMPLE III-1



```
CALL USTART
CALL UOUTLN
CALL UWINDO (-50000.,50000.,0.00001,0.00005)
CALL UMOVE (-40000.,0.00004)
CALL USET ('ARROW')
CALL UPEN (40000.,0.00004)
CALL UPEN (40000.,0.00002)
CALL UMOVE (-40000.,0.00004)
CALL USET ('DARROW')
CALL UPEN (40000.,0.00002)
CALL UEND
STOP
END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE ELEMENTARY SIX-LEVEL
C     ZOOMING BY ADJUSTING ONLY THE VIRTUAL WINDOW BOUNDARIES. NOTE
C     THAT THE PEN COMMANDS REQUIRED TO DRAW THE FIGURE REMAIN
C     UNCHANGED.
C
C     ENTER GCS AND SETUP LOOP TO PERMIT US TO ZOOM AWAY FROM FIGURE.
C
      CALL USTART
      DO 1 I=1,6
C
C     ERASE THE SCREEN AND DEFINE THE BOUNDARIES FOR OUR NEW WINDOW.
C
      CALL UERASE
      BOUNDS=100.*FLOAT(I)
      CALL UWINDO (0.,BOUNDS,0.,BOUNDS)
C
C     OUTLINE THE DEFAULT DEVICE PLOTTING AREA, AND DRAW THE FIGURE.
C
      CALL UOUTLN
      CALL DRWFIG
    1 CONTINUE
C
C     WRAP-UP ALL GRAPHIC ACTIVITY AND TERMINATE THE FORTRAN
C     PROGRAM.
C
      CALL UEND
      STOP
      END
      SUBROUTINE DRWFIG
C
C     SUBROUTINE USED TO GENERATE A PENTAGON WITHIN A CIRCLE & PAUSE.
C     SEVERAL GCS UTILITY SUBROUTINES ARE UTILIZED BUT NOT DESCRIBED
C     AT THIS POINT. (SEE CHAPTERS V AND VI FOR DETAILS).
C
      CALL UCRCLE (50.,50.,50.)
      CALL UPLYGN (50.,50.,5.,25.)
      CALL UBELL
      CALL UPAUSE
      RETURN
      END
```

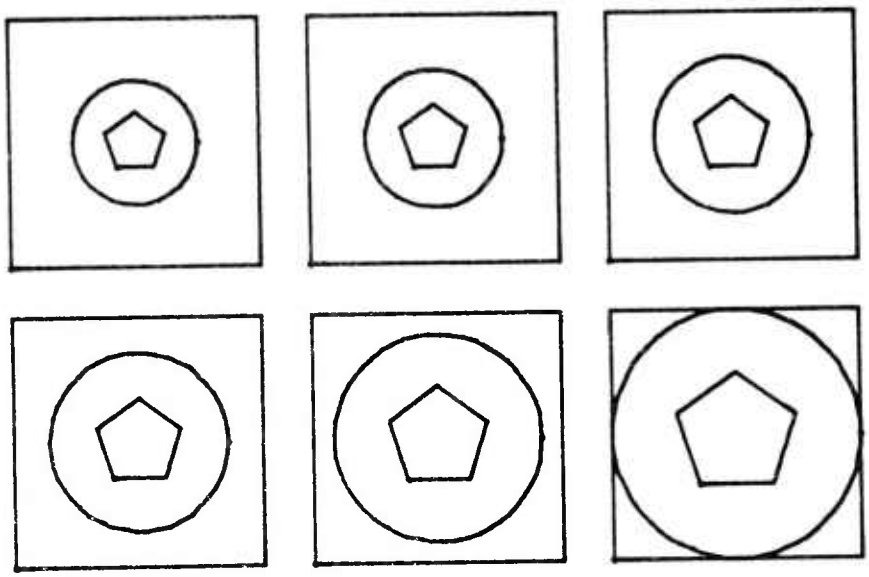
EXAMPLE III-2



```
      CALL USTART
      DO 1 I = 1, 6
      CALL UERASE
      BOUNDS = 100. * FLOAT(I)
      CALL UWINDO (0.,BOUNDS,0.,BOUNDS)
      CALL UOUTLN
      CALL DRWFIG
    1 CONTINUE
      CALL UEND
      STOP
      END
      SUBROUTINE DRWFIG
      CALL UCRCLE (50.,50.,50.)
      CALL UPLYGN (50.,50.,5.,25.)
      CALL UBELL
      CALL UPAUSE
      RETURN
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE ELEMENTARY SIX-LEVEL
C       ZOOMING BY ADJUSTING ONLY THE VIRTUAL WINDOW BOUNDARIES. NOTE
C       THAT THE PEN COMMANDS REQUIRED TO DRAW THE FIGURE REMAIN
C       UNCHANGED.
C
C       ENTER GCS AND SETUP A LOOP TO PERMIT US TO ZOOM TOWARD FIGURE.
C
        CALL USTART
        DO 1 I=1,6
C
C       ERASE THE SCREEN AND DEFINE THE BOUNDARIES FOR OUR NEW WINDOW.
C
        CALL UERASE
        BOUNDS=50.-(5.*FLOAT(I-1))
        CALL UWINDO (-BOUNDS,BOUNDS,-BOUNDS,BOUNDS)
C
C       OUTLINE THE DEFAULT DEVICE PLOTTING AREA, AND DRAW THE FIGURE.
C
        CALL UOUTLN
        CALL DRWFIG
    1   CONTINUE
C
C       WRAP-UP ALL GRAPHIC ACTIVITY AND TERMINATE THE FORTRAN
C       PROGRAM.
C
        CALL UEND
        STOP
        END
        SUBROUTINE DRWFIG
C
C       SUBROUTINE USED TO GENERATE A PENTAGON WITHIN A CIRCLE & PAUSE.
C       SEVERAL GCS UTILITY SUBROUTINES ARE UTILIZED BUT NOT DESCRIBED
C       AT THIS POINT. (SEE CHAPTERS V AND VI FOR DETAILS).
C
        CALL UCRCLE (0.,0.,25.)
        CALL UPLYGN (0.,0.,5.,12.5)
        CALL UBELL
        CALL UPAUSE
        RETURN
        END
```

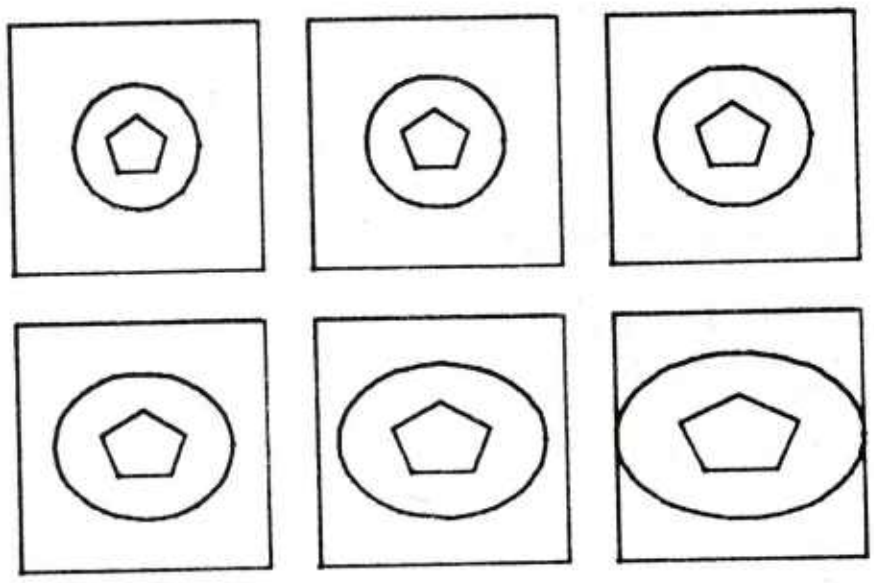
EXAMPLE III-3



```
      CALL USTART
      DO 1 I = 1, 6
      CALL UERASE
      BOUNDS = 50. - (5. * FLOAT(I-1))
      CALL UWINDO (-BOUNDS,BOUNDS,-BOUNDS,BOUNDS)
      CALL UOUTLN
      CALL DRWFIG
    1 CONTINUE
      CALL UEND
      STOP
      END
      SUBROUTINE DRWFIG
      CALL UCRCLE (0.,0.,25.)
      CALL UPLYGN (0.,0.,5.,12.5)
      CALL UBELL
      CALL UPAUSE
      RETURN
      END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE ELEMENTARY SIX-LEVEL
C     ZOOMING BY ADJUSTING ONLY THE VIRTUAL WINDOW BOUNDARIES. NOTE
C     THAT THE PEN COMMANDS REQUIRED TO DRAW THE FIGURE REMAIN
C     UNCHANGED. ALSO NOTE THE DISTORTION DUE TO THE EFFECT OF NON-
C     SQUARE WINDOWING.
C
C     ENTER GCS AND SETUP A LOOP TO PERMIT US TO ZOOM TOWARD FIGURE.
C
      CALL USTART
      DO 1 I=1,6
C
C     ERASE THE SCREEN AND DEFINE THE BOUNDARIES FOR OUR NEW WINDOW
C
      CALL UERASE
      XBOUND=50.-(5.*FLOAT(I-1))
      YBOUND=50.-(2.5*FLOAT(I-1))
      CALL UWINDO (-XBOUND,XBOUND,-YBOUND,YBOUND)
C
C     OUTLINE THE DEFAULT DEVICE PLOTTING AREA WE HAVE JUST DEFINED,
C     THEN DRAW THE FIGURE.
C
      CALL UOUTLN
      CALL DRWFIG
    1 CONTINUE
C
C     WRAP-UP ALL GRAPHIC ACTIVITY AND TERMINATE THE FORTRAN PROGRAM
C
      CALL UEND
      STOP
      END
      SUBROUTINE DRWFIG
C
C     SUBROUTINE USED TO GENERATE A PENTAGON WITHIN A CIRCLE & PAUSE.
C     SEVERAL GCS UTILITY SUBROUTINES ARE UTILIZED BUT NOT DESCRIBED
C     AT THIS POINT. (SEE CHAPTERS V AND VI FOR DETAILS).
C
      CALL UCRCLE (0.,0.,25.)
      CALL UPLYGN (0.,0.,5.,12.5)
      CALL UBELL
      CALL UPAUSE
      RETURN
      END
```

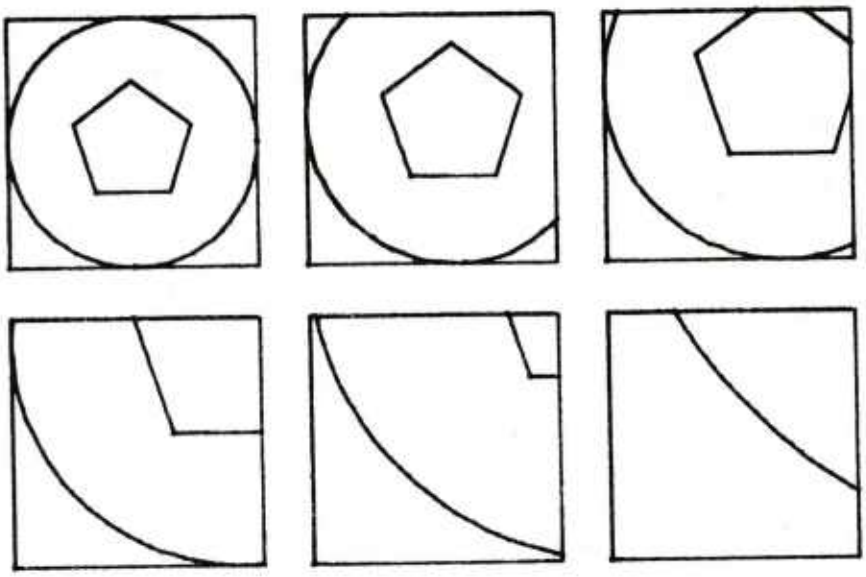
EXAMPLE III-4



```
      CALL UGSTART
      DO 1 I = 1, 6
      CALL UERASE
      XBOUND = 50. - (5. * FLOAT(I-1))
      YBOUND = 50. - (2.5 * FLOAT(I-1))
      CALL UWINDO (-XBOUND, XBOUND,-YBOUND,YBOUND)
      CALL UOUTLN
      CALL DRWFIG
    1 CONTINUE
      CALL UEND
      STOP
      END
      SUBROUTINE DRWFIG
      CALL UCRCLE (0.,0.,25.)
      CALL UPLYGN (0.,0.,5.,12.5)
      CALL UBELL
      CALL UPAUSE
      RETURN
      END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE ELEMENTARY SIX-LEVEL
C     ZOOMING BY ADJUSTING ONLY THE VIRTUAL WINDOW BOUNDARIES. NOTE
C     THAT THE PEN COMMANDS REQUIRED TO DRAW THE FIGURE REMAIN
C     UNCHANGED. ALSO NOTE THE CLIPPING OF THE DISPLAY AT THE WINDOW
C     BOUNDARY.
C
C     ENTER GCS AND SETUP A LOOP TO PERMIT US TO ZOOM TOWARD FIGURE.
C
C     CALL USTART
C
      DO 1 I=1,6
C
C     ERASE THE SCREEN AND DEFINE THE BOUNDARIES FOR OUR NEW WINDOW.
C
      CALL UERASE
      BOUNDS=100.-(15.*FLOAT(I-1))
      CALL UWINDO (0.,BOUNDS,0.,BOUNDS)
C
C     OUTLINE THE DEFAULT DEVICE PLOTTING AREA WE HAVE JUST DEFINED,
C     THEN DRAW THE FIGURE.
C
      CALL UOUTLN
      CALL DRWFIG
    1 CONTINUE
C
C     WRAP-UP ALL GRAPHIC ACTIVITY AND TERMINATE THE FORTRAN
C     PROGRAM.
C
      CALL UEND
      STOP
      END
      SUBROUTINE DRWFIG
C
C     SUBROUTINE USED TO GENERATE A PENTAGON WITHIN A CIRCLE AND
C     PAUSE. SEVERAL GCS UTILITY SUBROUTINES ARE UTILIZED BUT NOT
C     DESCRIBED AT THIS POINT. (SEE CHAPTERS V AND VI FOR DETAILS).
C
      CALL UCRCLE (50.,50.,50.)
      CALL UPLYGN (50.,50.,5.,25.)
      CALL UBELL
      CALL UPAUSE
      RETURN
      END
```

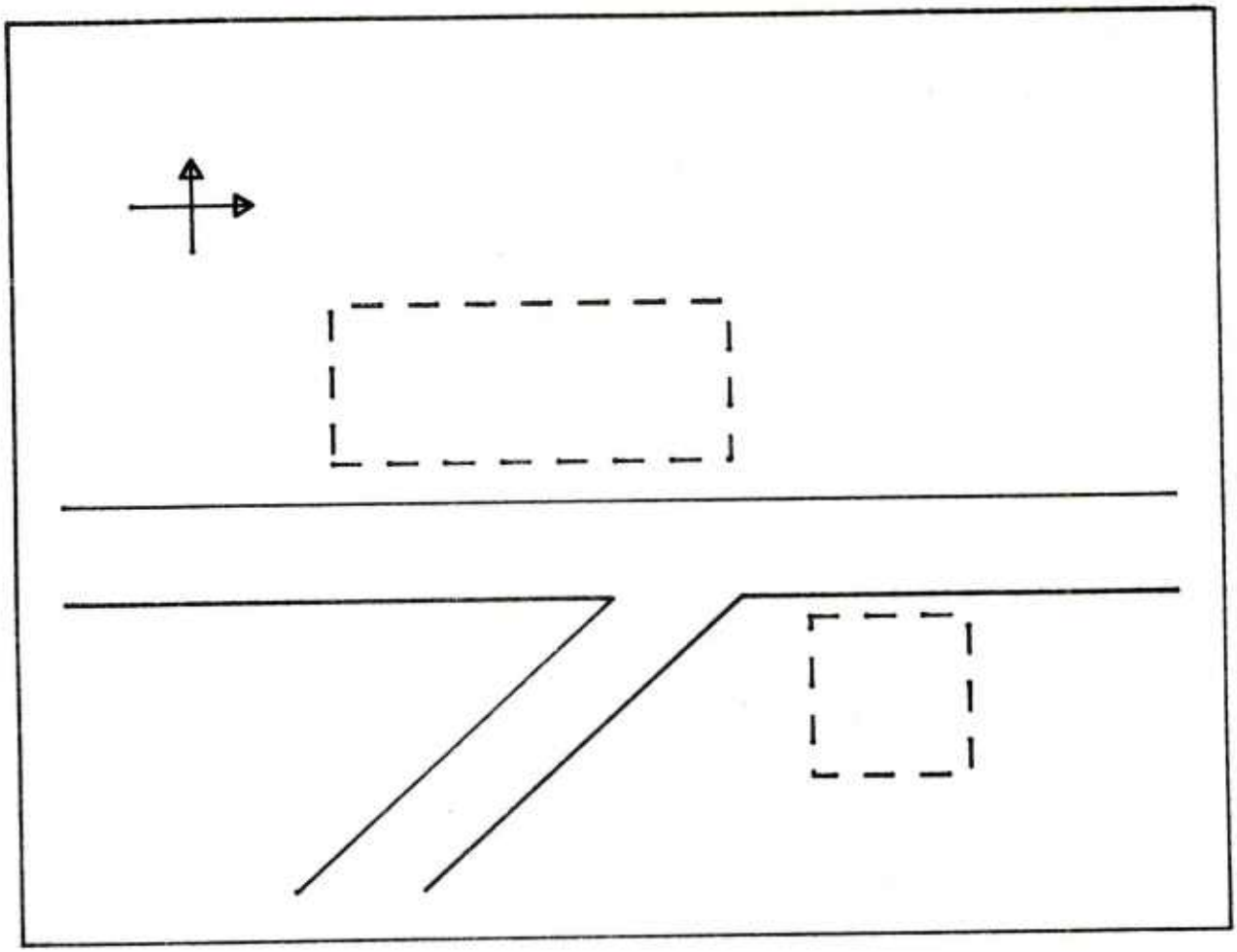
EXAMPLE III-5



```
      CALL USTART
      DO 1 I = 1, 6
      CALL UERASE
      BOUNDS = 100. - (15. * FLOAT(I-1))
      CALL UWINDO (0.,BOUNDS,0.,BOUNDS)
      CALL UOUTLN
      CALL DRWFIG
    1 CONTINUE
      CALL UEND
      STOP
      END
      SUBROUTINE DRWFIG
      CALL UCRCLE (50.,50.,50.)
      CALL UPLYGN (50.,50.,5.,25.)
      CALL UBELL
      CALL UPAUSE
      RETURN
      END
```

```
C     THIS PROGRAM GENERATES A SIMPLE STREET DIAGRAM FOR A TEKTRONIX
C     4010/4013 TERMINAL.
C
C     INITIALIZATION, DEVICE MODE ENTRY, AND OUTLINE GENERATION
C
      CALL USTART
      CALL USET ('DEVICE')
      CALL UOUTLN
C
C     GENERATION OF ROADS WITH THE DEFAULT CASE OF LINES IN TERMS OF
C     INCHES.
C
      CALL UMOVE (0.3,2.7)
      CALL UPEN (7.2,2.7)
      CALL UMOVE (7.2,2.1)
      CALL UPEN (4.5,2.1)
      CALL UPEN (2.5,0.3)
      CALL UMOVE (1.7,0.3)
      CALL UPEN (3.7,2.1)
      CALL UPEN (0.3,2.1)
C
C     GENERATION OF HOUSES WITH DASHED LINES IN TERMS OF CENTIMETERS.
C
      CALL USET ('CENTIMETERS')
      CALL USET ('DASH')
      CALL UMOVE (12.5,5.0)
      CALL UPEN (15.5)
      CALL UPEN (15.2.5)
      CALL UPEN (12.5.2.5)
      CALL UPEN (12.5,5)
      CALL UMOVE (5.,7.5)
      CALL UPEN (11.3,7.5)
      CALL UPEN (11.3,10.0)
      CALL UPEN (5.,10.)
      CALL UPEN (5.,7.5)
C
C     GENERATION OF DIRECTION REFERENCES WITH ARROW LINES IN TERMS OF
C     PERCENTUNITS
C
      CALL USET ('PERCENTUNITS')
      CALL USET ('ARROW')
      CALL UMOVE (10.,80)
      CALL UPEN (20.,80.)
      CALL UMOVE (15.,75.)
      CALL UPEN (15.,85.)
C
C     TERMINATION
C
      CALL UEND
      STOP
      END
```

EXAMPLE III-6

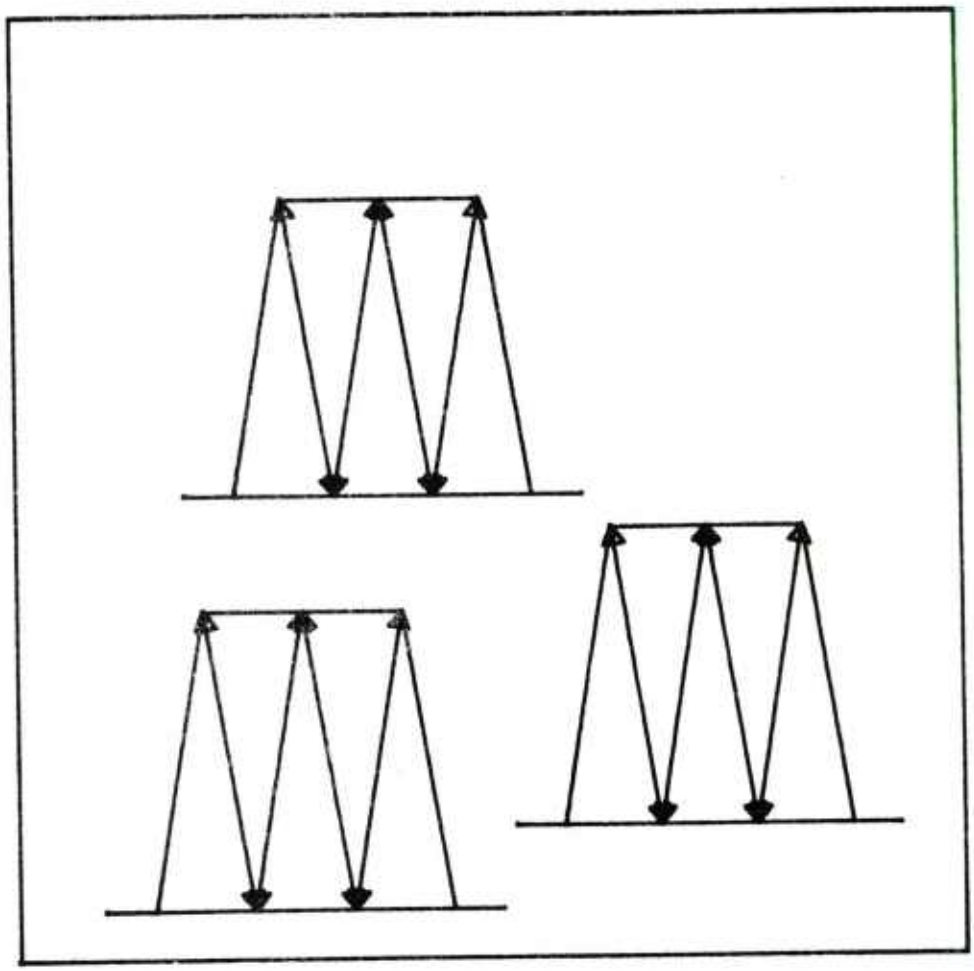


```
CALL USTART
CALL USET ('DEVICE')
CALL UOUTLN
CALL UMOVE (0.3,2.7)
CALL UPEN (7.2,2.7)
CALL UMOVE (7.2,2.1)
CALL UPEN (4.5,2.1)
CALL UPEN (2.5,0.3)
CALL UMOVE (1.7,0.3)
CALL UPEN (3.7,2.1)
CALL UPEN (0.3,2.1)
CALL USET ('CENTIMETERS')
CALL USET ('DASH')
CALL UMOVE (12.5,5.0)
CALL UPEN (15.,5.)
CALL UPEN (15.,2.5)
CALL UPEN (12.5,2.5)
CALL UPEN (12.5,5.)
CALL UMOVE (5.,7.5)
CALL UPEN (11.3,7.5)
CALL UPEN (11.3,10.0)
CALL UPEN (5.,10.)
CALL UPEN (5.,7.5)
CALL USET ('PERCENTUNITS')
CALL USET ('ARROW')
CALL UMOVE (10.,80.)
CALL UPEN (20.,80.)
CALL UMOVE (15.,75.)
CALL UPEN (15.,85.)
CALL UEND
STOP
END
```

```
C       THIS PROGRAM GENERATES THE SAME DISPLAY AT VARIOUS LOCATIONS
C       ON A TEKTRONIX 4010/4013 TERMINAL.
C
C       INITIALIZATION AND OUTLINE.
C
        CALL USTART
        CALL UOUTLN

C       FOR EACH OF 3 PASSES, DEFINE A DEVICE PLOTTING AREA AND 'DRAW' THE
C       GRAPH.
C
        DO 10 I=1,3
        IF(I.EQ.1)CALL UDAREA (2.,5.,0.,3.)
        IF(I.EQ.2)CALL UDAREA (4.5,7.4,0.5,3.5)
        IF(I.EQ.3)CALL UDAREA (2.5,5.5,2.5,5.5)
C
C       CALL SUBROUTINE TO DRAW GRAPH.
C
   10   CONTINUE
C
C       TERMINATION
C
        CALL UEND
        STOP
        END
C
C       SUBROUTINE USED TO DRAW GRAPH.
C
        SUBROUTINE GRAFIT
        CALL UMOVE (10.,10.,10.)
        CALL USET ('LINE')
        CALL UPEN (90.,10.)
        CALL UMOVE (20.,10.)
        CALL UPEN (30.,70.)
        CALL UPEN (70.,70.)
        CALL UPEN (80.,10.)
        CALL UMOVE (30.,70.)
        CALL USET ('DOUBLEARROW')
        CALL UPEN (40.,10.)
        CALL UPEN (50.,70.)
        CALL UPEN (60.,10.)
        CALL UPEN (70.,70.)
        RETURN
        END
```

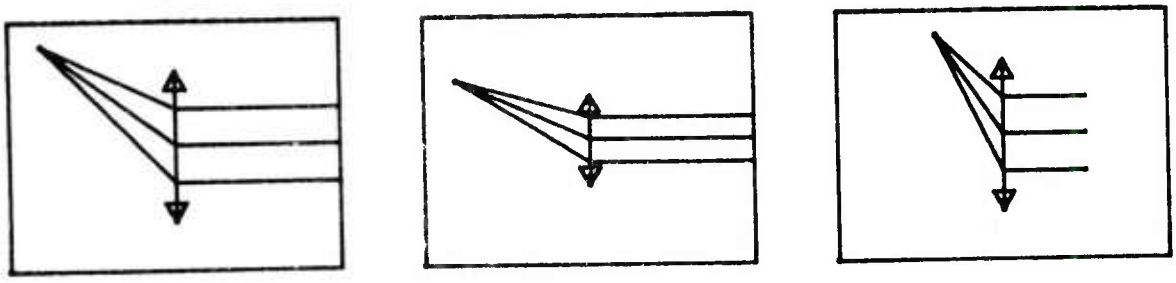
EXAMPLE III-7



```
      CALL USTART
      CALL UOUTLN
      DO 10 I = 1, 3
      IF (I .EQ. 1) CALL UDAREA (2.,5.,8.,9.)
      IF (I .EQ. 2) CALL UDAREA (4.5,7.4,8.5,9.5)
      IF (I .EQ. 3) CALL UDAREA (2.5,5.5,2.5,5.5)
      CALL GRAFIT
   10 CONTINUE
      CALL UEND
      STOP
      END
      SUBROUTINE GRAFIT
      CALL UMOVE (10.,10.)
      CALL USET ('LINE')
      CALL UPEN (90.,10.)
      CALL UMOVE (20.,10.)
      CALL UPEN (30.,70.)
      CALL UPEN (70.,70.)
      CALL UPEN (80.,10.)
      CALL UMOVE (30.,70.)
      CALL USET ('DOUBLEARROW')
      CALL UPEN (40.,10.)
      CALL UPEN (50.,70.)
      CALL UPEN(60.,10.)
      CALL UPEN (70.,70.)
      RETURN
      END
```

```
C       THIS PROGRAM GENERATES A DISPLAY, A VERTICALLY DISTORTED
C       VERSION OF THE DISPLAY, AND A HORIZONTALLY DISTORTED VERSION OF
C       THE DISPLAY FOR A TEKTRONIX 4010/4013 TERMINAL.
C
C       INITIALIZATION.
C
        CALL USTART
C
C       FOR EACH OF 3 PASSES, ERASE THE SCREEN, PROVIDE AN OUTLINE, DEFINE
C       THE DEVICE WINDOW, AND 'DRAW' THE GRAPH.
C
        DO 10 I=1,3
        CALL UERASE
        CALL UDAREA (0.,7.5,0.,5.5)
        CALL UOUTLN
        IF(I.EQ.2)CALL UDAREA (0.,7.5,2.4,4.)
        IF(I.EQ.3)CALL UDAREA (3.,5.,0.,5.5)
        CALL UMOVE (50.,20.)
        CALL UPEN1 (50.,80.,'DOUBLEARROW')
        CALL UMOVE (10.,90.)
        CALL UPEN (50.,65.)
        CALL UPEN (100.,65)
        CALL UMOVE (10.,90.)
        CALL UPEN (50.,50.)
        CALL UPEN (100.,50.)
        CALL UMOVE (10.,90.)
        CALL UPEN (50.,35.)
        CALL UPEN (100.,35.)
   10   CONTINUE
C
C       TERMINATION.
C
        CALL UEND
        STOP
        END
```

EXAMPLE III-8



```
      CALL USTART
      DO 10 I = 1, 3
      CALL UERASE
      CALL UDAREA (0.,7.4,8.,5.5)
      CALL UOUTLN
      IF (I .EQ. 2) CALL UDAREA (0.,7.4,2.,4.)
      IF (I .EQ. 3) CALL UDAREA (3.,5.,8.,5.5)
      CALL UMOVE (50.,20.)
      CALL UPEN1 (50.,80.,'DOUBLEARROW')
      CALL UMOVE (10.,90.)
      CALL UPEN (50.,85.)
      CALL UPEN (100.,85.)
      CALL UMOVE (10.,90.)
      CALL UPEN (50.,50.)
      CALL UPEN (100.,50.)
      CALL UMOVE (10.,90.)
      CALL UPEN (50.,35.)
      CALL UPEN (100.,35.)
   10 CONTINUE
      CALL UEND
      STOP
      END
```

```
C     THIS PROGRAM DEMONSTRATES HOW TO SUPPORT DEVICE
C     INDEPENDENCE.
C
      DIMENSION ARRAY (8)
C
C     INITIALIZE
C
      CALL USTART
C
C     SET 'DEVICE' UNITS TO 'PERCENTUNITS' and call UDAREA for 100%. SINCE
C     ONE PERCENTUNIT IN THE X MAY DIFFER FROM ONE PERCENTUNIT IN THE Y,
C     RESET TO 'INCHES' AND THEN DIVIDE SCREEN.
C
      CALL USET ('PERCENTUNITS')
      CALL UDAREA (0.,100.,0.,100.)
      CALL USET ('INCHES')
      CALL USTUD ('ARRAY')
C
C     ASSUME SCREEN IS LONGER IN X
C
      XMIN = ARRAY(5)
      XMAX = AMIN1(ARRAY(6)/2.,ARRAY(8))
      YMIN = ARRAY(7)
      YMAX = XMAX
C
C     PLOT THE SAME FIGURE ON HALF OF THE SCREEN
C
      CALL UDAREA (XMIN,XMAX,YMIN,YMAX)
      CALL UOUTLN
      CALL FIGURE
      CALL UDAREA (XMAX,XMAX*2.,YMIN,YMAX)
      CALL UOUTLN
      CALL FIGURE
C
C     TERMINATION
C
      CALL UEND
      STOP
      END
C
C     SUBROUTINE TO GENERATE A POLYGON
C
      SUBROUTINE FIGURE
      CALL UPLYGN (50.,50.,5.,25.)
      RETURN
      END
```

EXAMPLE III-9

CHAPTER IV

ALPHANUMERIC OUTPUT

Basic Alphanumeric Output

Under many circumstances, the graphics programmer would prefer to manipulate alphanumeric data with the same ease that pertains to strictly graphical information. Such a capability would expedite the labeling of figures, allow numerical values to be printed at a specific location on the display, and permit a discrete separation of text from graphical information should the user so desire. In order to provide these capabilities, GCS has incorporated two very powerful alphanumeric output and editing routines, UPRINT and UWRITE, which may be called by the following sequences:

```
CALL UPRINT (X,Y,DATA)
CALL UWRITE (X,Y,DATA)
```

X and Y are used to specify the starting position (in current units) of the first character of the output text. All positional and coordinate addressing related to characters is assumed to specify the position of the lower-left corner of the given character. DATA is interpreted in the following manner, based upon one of the four options available to the user through a call to USET before invoking UPRINT or UWRITE:

A. **TEXT**—*Under this default option, the routines will assume that DATA contains Hollerith information delimited by a terminator and will output all text up to, but not including that delimiter, beginning at the location specified by (X,Y). CALL UPSET ('TERMINATOR', CHARACTER) can change the terminator from the default backslash character. There are four 'hardware' character sizes that can be changed by CALL USET(SIZE) where SIZE can be 'EXTRALARGE', 'LARGE', 'MEDIUM', or 'SMALL' (default).*

B. **REALNUMBER**—*DATA is assumed to be a single-valued parameter whose contents is assumed to be of type REAL. Editing will take place using 'G' format, and the text will be printed beginning at (X,Y).*

C. **INTEGER**—*As in the case of REALNUMBER, DATA is assumed to be single-valued and of type REAL. The routines will perform a REAL to INTEGER conversion and will print the result in 'I' format.*

D. **XYCOORDINATES**—*DATA is assumed to be a 2 element REAL array whose values specify two numbers to be editted and printed in the form: (X,Y).*

UPRINT and UWRITE perform identical functions and differ only in the action taken upon return from execution of the given subroutine. Upon exit, UWRITE moves the beam position back to its original coordinates at the time it was invoked, whereas UPRINT leaves the beam position at the end of the output text. Example IV-1 illustrates the four options above.

Alphanumeric output through GCS is perhaps most conveniently manipulated through the use of FONTUNIT coordinates under DEVICE mode. Under this USET option, the (X,Y) coordinates are expressed directly in number of characters horizontally or vertically.

Margining

Since alphanumeric output through UPRINT and UWRITE can be 'clipped' when in 'VIR-

TUAL' mode the user is provided the capability to horizontally and vertically margin his output (when in device mode) through an invocation to UMARGN:

```
CALL UMARGN (XLEFT,XRIGHT,YBOTTM,YTOP)
```

XLEFT and XRIGHT represent the desired left-most and right-most horizontal positions expressed in DEVICE units within which alphanumeric output may be permitted. Maximum and minimum vertical margins are established by YBOTTM and YTOP respectively. Should values of XLEFT, XRIGHT, YTOP, and YBOTTM be specified such that XRIGHT < XLEFT or YTOP < YBOTTM, UMARGN will ignore the specified setting and the margins will reflect their values prior to the call to UMARGN.

Margining affects alphanumeric output (when in device mode) in the following manner:

A. Should any character extend beyond the right margin, an automatic carriage return and line feed are generated, followed by a position to the left margin and output resumed.

B. Should the user position the beam at a location to the left of the left margin, the beam is positioned to the left margin before output is initiated.

C. Should output be attempted at vertical positions outside the closed interval defined by the minimum and maximum vertical margins, the beam is moved to the maximum vertical margin before any output is begun.

See Example IV-2 for an example of margining.

Bulk or Mixed Alphanumeric Output

For applications requiring bulk or mixed alphanumeric output, the graphics programmer should consider the use of subroutine UPRNT1 and UWRIT1, which may be invoked by the following calling sequence:

```
CALL UPRNT1 (DATA,OPTION)
CALL UWRIT1 (DATA,OPTION)
```

OPTION is a single-valued character variable which specifies the format or mode under which DATA is to be editted and displayed; i.e., TEXT, REALNUMBER, GREEN, etc. DATA specifies the actual information which is to be output by UPRNT1 and UWRIT1 under the given OPTION. It should be noted that the constraints on DATA are identical to those imposed by UPRINT and UWRITE.

UPRNT1 and UWRIT1 differ from UPRINT and UWRITE in the following respects:

A. Since no coordinate specifications are passed in the calling sequence, the alphanumeric output will begin at the current beam position upon entry to the subroutine.

B. The effects of OPTION apply only to the current output operation and upon exit from the subroutine these effects are removed from the GSA.

Fortran Input-Output

For alphanumeric output used in interactive input-output communications and various control and coordination activities (such as the printout of warning or error notifications), the convenience of permitting the use of normal FORTRAN input-output statements may

override the advantages of using UPRINT or UWRITE to process text. For such activities to be performed successfully, it is essential that the terminal device be set for the receipt of alphanumeric rather than graphic information, a task which is accomplished through a call to **UALPHA.** On systems that buffer the graphics output, **UALPHA** must be called prior to the Fortran I/O to flush the graphics output. It should be noted that all alphanumeric FORTRAN output is unaffected by the margins established by the user.

Single Character Output

In isolated instances, it is desirable to combine the output of graphical and alphanumeric information into one composite operation. This capability is provided through the 'CHARACTER' option available with UPEN, whereby a single printable character used as a parameter to USET identifies that parameter as the current character for printing. Thus, in order to print the character 'A' at (X,Y), the following commands would be specified:

```
CALL USET ('LA')
CALL UPEN (X,Y)
```

Upon execution of the call to UPEN, the character 'A' will terminate the line drawn under the current line option. It should be noted that the CHARACTER mode of operation used in the above manner will remain in effect until any one of the acceptable UPEN suffixes is specified; e.g., CALL USET ('LNULL'). See Example IV-3.

An additional facility for single-character output is provided through subroutine UAOUT which is called in the following manner:

```
CALL UAOUT (CHAR)
```

CHAR is a single character expressed in Hollerith format either as a quoted character string or as an explicit variable. Since UAOUT outputs the given character at the current beam position, the user should ensure that the beam is positioned to the proper coordinates before invoking UAOUT. Character output through UAOUT is subject to the current margin constraints applicable to UPRINT and UWRITE: in addition, this output differs from that available through UPEN in that UAOUT positions the beam to next character position after the specified character has been printed, whereas the UPEN option always maintains the beam position at the center of the desired character. See Example IV-4 and compare to Example IV-3.

Software Character Output

The preceding discussion of alphanumeric output has been applicable only to HARDWARE character generation. Under certain circumstances, however, the sophisticated user may desire character output which is rotated or scaled to suit a particular application. For examples see Example IV-5, and IV-6. This capability is provided in GCS through the SOFTWARE character option available through USET. Under this option, all output is subject to the constraints imposed by the virtual window, whenever UPRINT is called in 'VIRTUAL' space. Although SOFTWARE characters are manipulated using the same routines as those for output of HARDWARE characters, interested users are directed to the relevant subroutine writeups for a discussion of the limitations imposed on this form of output.

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE OPTIONS AVAILABLE THROUGH
C     'UPRINT' AND 'UWRITE'.  DEFAULT VIRTUAL WINDOW AND DEVICE AREA WILL
C     BE USED.
C
C     FIRST ESTABLISH A 2-COORDINATE ARRAY:  COORD(2). PUT SAMPLE DATA
C     POINT (25.,0.9999999E19) IN ARRAY.  NOTE THAT COORDINATES SPECIFIED
C     IN ARRAY COORD ARE PRINTED.  THEY MAY OR MAY NOT BE SAME AS
C     COORDINATES WHICH SPECIFY WHERE PRINTING IS TO OCCUR.
C
      DIMENSION COORD (2)
      DATA COORD/25.,0.9999999E19/
C
C     DEFAULT OPTION FOR 'UPRINT' AND 'UWRITE' IS FOR 'TEXT'.  USE THIS
C     DEFAULT TO OUTPUT A LINE OF TEXT AT SAMPLE COORDINATE LOCATION
C     (0.,100.) NOTE THE SEMICOLAN(;) IS THE DELIMITER.
C
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL UPRINT (0.,100.,'THIS IS A SAMPLE LINE OF OUTPUT TEXT;')
C
C     SPECIFY THE 'REALNUMBER' OF OPERATION AND USE UWRITE TO PRINT AT
C     MIDSCREEN (50.,50.) A TYPICAL REAL NUMBER (100.)
C
      CALL USET ('REALNUMBER')
      CALL UWRITE (50.,50.,100.)
C
C     SPECIFY 'INTEGER' MODE AND PRINT AT COORDINATES (75.,25.) THE
C     SAMPLE INTEGER -123456789  NOTICE THAT SINCE ALL GCS PARAMETERS
C     ARE REAL NUMBERS, EVEN THIS INTEGER MUST BE PASSED IN REAL
C     NUMBER FORM I.E. -123456789.
C
      CALL USET ('INTEGER')
      CALL UPRINT (75.,25.,-123456789.)
C
C     EXAMPLE OF USING 'XYCOORDINATE' OPTION TO PRINT XYCOORDINATES.
C     NOTE THE VARIED FORM OF OUTPUT OF REAL NUMBERS WITH 'G' FORMAT
C
C
      CALL USET ('XYCOORDINATE')
      CALL UWRITE (25.,75.,COORD)
C
C     END GCS (CALL UEND); STOP EXECUTION (STOP), END PROGRAM (END)
C
      CALL UEND
      STOP
      END
```

EXAMPLE IV-1

THIS IS A SAMPLE LINE OF OUTPUT TEXT

(25.,.1000E+20)

100.

-123456789

```
DIMENSION COORD (2)
DATA COORD/25.,0.9999999E19/
CALL USTART
CALL UPSET ('TERMINATOR',';')
CALL UPRINT (0.,75.,'THIS IS A SAMPLE LINE OF OUTPUT TEXT;')
CALL USET ('REALNUMBER')
CALL UWRITE (50.,25.,100.)
CALL USET ('INTEGER')
CALL UPRINT (75.,0.,-123456789.)
CALL USET ('XYCOORDINATE')
CALL UWRITE (25.,50.,COORD)
CALL UEND
STOP
END
```

```
C       SAMPLE PROGRAM TO ILLUSTRATE OPTIONS AVAILABLE THROUGH
C       MARGINNING.  DEFAULT VALUES WILL BE USED IN THIS EXAMPLE WITH
C       ADDITIONAL CALLS TO UMARGN TO ADJUST THE ALPHANUMERIC WINDOW.
C       THIS PROGRAM WAS WRITTEN FOR A TEKTRONIX 4010/4013 TERMINAL.
C
C       SET UP 300 CHARACTER ARRAY NAMED SAMPLE; PUT TEXT IN IT.  NOTE
C       SEMICOLON (;) as character terminator

        CHARACTER SAMPLE*300

        DATA SAMPLE/'THIS IS A LINE OF OUTPUT TEXT WHICH IS LONG ENOUGH TO
     1  CAUSE THE ALPHANUMERIC OUTPUT TO WRAP-AROUND.  NOTE THE
     2  EFFECTS WHICH THE DEFAULT MARGINS HAVE UPON OUTPUT;'/INITIALIZE
C
        CALL USTART
C
C       NOW PRINT IT WITH FIRST CHARACTER OF PRINT STRING AT COORDINATE
C       LOCATION (20,25)
C
        CALL UPSET ('TERMINATOR', ';')
        CALL USET ('FONTUNITS')
        CALL UMARGN (35.,36.,1.,35.)
        CALL USET ('PERCENTUNITS')
        CALL USET ('DEVICE')
        CALL UPRINT (20.,25.,SAMPLE)
C
C       SECOND EXAMPLE
C
C       EXAMPLE ILLUSTRATING THE USEFULNESS OF MARGINING.  THE MARGINS
C       WILL BE SET USING 'FONTUNIT' COORDINATE ADDRESSING WITH THE LEFT
C       MARGIN AT APPROXIMATELY MID-SCREEN, I.E. 35 CHARACTER-WIDTHS OR
C       FONTUNITS FROM THE LEFT EDGE, AND THE RIGHT MARGIN 1 FONTUNIT
C       LATER AT POSITION 36, GIVING A TOTAL MARGIN WIDTH OF ONE
C       CHARACTER.
C
C
C       POSITION PRINTING AT TOP OF PAPER.  IN THIS EXAMPLE WE SPECIFY 4000
C       CHARACTER-HEIGHTS (FONTUNITS) FROM BOTTOM OF PAPER, AN
C       UNREASONABLY LARGE VALUE, SO GCS DEFAULTS TO THE LARGEST
C       PHYSICALLY POSSIBLE VALUE, THE TOP OF THE PAGE.
C
        CALL UPRINT (0.,4000.,'HELLO THERE!;')
C
C       WRAP UP
C
        CALL UEND
        STOP
        END
```

EXAMPLE IV-2

H
E
L
L
O

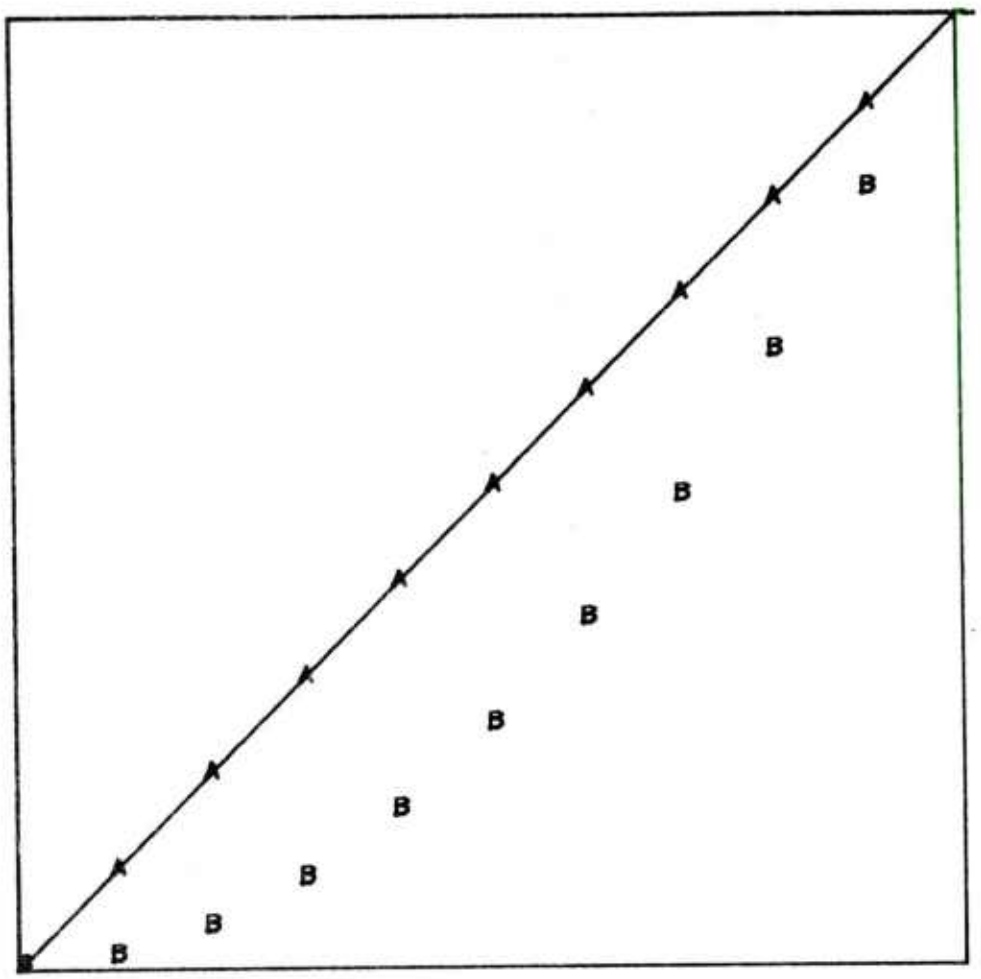
T
H
E
R
E
!

THIS IS A LINE OF OUTPUT TEXT WHICH IS LONG ENOUGH TO CAUS
E THE ALPHANUMERIC OUTPUT TO WRAP-AROUND. NOTE THE EFFECTS WHICH THE DEFA
ULT MARGINS HAVE UPON OUTPUT

```
      CHARACTER SAMPLE*300
      DATA SAMPLE/'THIS IS A LINE OF OUTPUT TEXT
     & WHICH IS LONG ENOUGH TO CAUSE THE ALPHANUMERIC
     & OUTPUT TO WRAP-AROUND. NOTE THE EFFECTS WHICH THE
     & DEFAULT MARGINS HAVE UPON OUTPUT;'
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL USET ('PERCENTUNITS')
      CALL USET ('DEVICE')
      CALL UPRINT (20.,25.,SAMPLE)
      CALL USET ('FONTUNITS')
      CALL UMARGN (35.,36.,1.,35.)
      CALL UPRINT (0.,4000.,'HELLO THERE!;')
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE SINGLE CHARACTER OUTPUT
C       AND LINE TERMINATOR OPTIONS AVAILABLE THROUGH 'UPEN'. TWO
C       GRAPHS WILL BE PLOTTED:
C          (1)  A LINEAR EQUATION (Y=X). THE DATA FOR PLOTTING IT WILL BE
C               PRESTORED WITH THE VALUES OF THE INDEPENDENT VARIABLE X IN
C               THE ARRAY X AND CORRESPONDING VALUES OF Y IN THE ARRAY Y.
C
C          (2)  A QUADRATIC EQUATION (Z=(.1*X)**2). THE DATA VALUES FOR THE
C               SAME VALUES OF THE INDEPENDENT VARIABLE X WILL BE STORED
C               IN THE Z ARRAY.
C
C       DEFAULT VIRTUAL WINDOW AND DEVICE AREA WILL BE USED. THE LINEAR
C       EQUATION WILL HAVE AN 'A' TERMINATOR AT THE END OF VISIBLE LINE
C       SEGMENT, WHEREAS THE QUADRATIC EQUATION WILL HAVE A 'B'
C       TERMINATOR FOR EACH INVISIBLE LINE SEGMENT.
C
C       FIRST SET UP X,Y, & Z ARRAYS. PRESTORE DATA IN THEM
C
        DIMENSION X(11),Y(11),Z(11)
        DATA X/0.,10.,20.,30.,40.,50.,60.,70.,80.,90.,100./
        DATA Y/0.,10.,20.,30.,40.,50.,60.,70.,80.,90.,100./
        DATA Z/0.,1.,4.,9.,16.,25.,36.,49.,64.,81.,100./
C
C       ENTER GCS AND DRAW OUTLINE OF DEFAULT DEVICE PLOTTING AREA
C
        CALL USTART
        CALL UOUTLN
C
C       USTART HAS IMPLICITLY SET PEN TO SOLID-LINE MODE. NOW SPECIFY
C       THAT AN 'A' TERMINATOR SHOULD GO ON EVERY LINE AND MOVE TO INITIAL
C       (X,Y) POINT TO BE PLOTTED
C
        CALL USET ('LA')
        CALL UMOVE (X(1),Z(1))
C
C       DRAW 11 LINE SEGMENTS X(1),Y(1) TO X(2),Y(2) TO X(3),Y(3) AND SO ON TO
C       X(11),Y(11) THEREBY PLOTTING TOTAL EQUATION
C
        DO 1 I=1,11
    1   CALL UPEN (X(I),Y(I))
C
C       SET LINE TYPE TO 'NOLINE' AND SPECIFY THAT A 'B' TERMINATOR IS TO BE
C       USED FOR EACH INVISIBLE LINE SEGMENT
C
        CALL USET ('NB')
C
C       MOVE TO FIRST (X,Z) POINT, THEN PLOT 11 (X,Z) VALUES
C
        CALL UMOVE (X(1),Z(1)
        DO 2 I=11
    2   CALL UPEN (X(I),Z(I))
C
C       WRAP UP
C
        CALL UEND
        STOP
        END
```

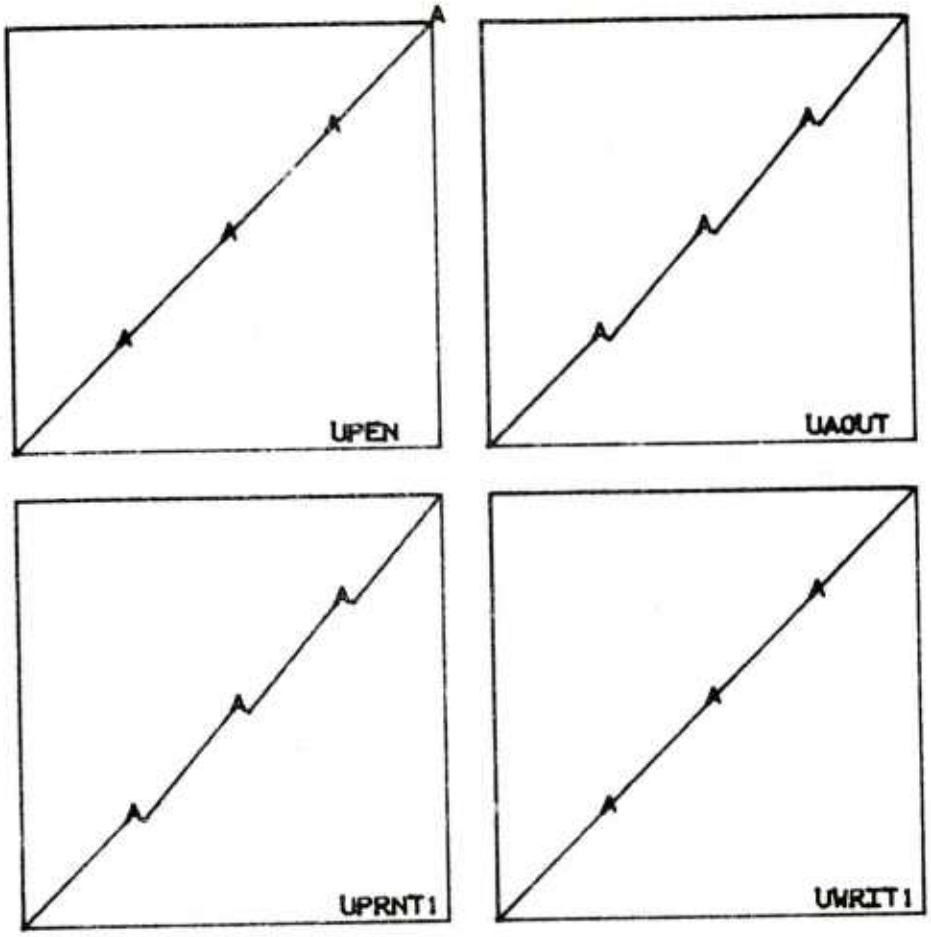
EXAMPLE IV-3




```
      DIMENSION X(11), Y(11), Z(11)
      DATA X/0.,10.,20.,30.,40.,50.,60.,70.,80.,90.,100./
      DATA Y/0.,10.,20.,30.,40.,50.,60.,70.,80.,90.,100./
      DATA Z/0.,1.,4.,9.,16.,25.,36.,49.,64.,81.,100./
      CALL USTART
      CALL UOUTLN
      CALL USET ('LA')
      CALL UMOVE (X(1),Z(1))
      DO 1 I = 1, 11
    1 CALL UPEN (X(I),Y(I))
      CALL USET ('NB')
      CALL UMOVE (X(1),Z(1))
      DO 2 I = 1, 11
    2 CALL UPEN (X(I),Z(I))
      CALL UEND
      STOP
      END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE CHARACTER TERMINATOR AND
C     A/N OUTPUT USED IN CONJUNCTION WITH GRAPHICAL OUTPUT FOR A
C     TEKTRONIX 4010/4013 TERMINAL.
C
      CHARACTER ROUTINE*7(4)
      DATA INDEX,YO/0,5.73/
      DATA ROUTINE/'UPEN;','UAOUT;','UPRINT;','UWRIT1;'/
C
C     ENTER GCS, DIVIDE PLOTTING AREA INTO 4 EQUAL SEGMENTS, CHOOSE
C     ONE OF THESE SEGMENTS, AND OUTLINE IT.
C
      CALL USTART
      CALL USET ('TERMINATOR', ';')
      DO 5 I=1,2
      XO=-1.82
      YO=YO-2.86
      DO 5 J=1,2
      XO=XO+2.86
      INDEX=INDEX+1
      CALL UDAREA (XO,(XO+2.57),YO,(YO+2.57))
      CALL UOUTLN
C
C     MOVE TO THE ORIGIN & SPECIFY STANDARD LINE WITH NO TERMINATOR. IF
C     'UPEN' OPTION IN EFFECT, SPECIFY AN 'A' AS THE TERMINATOR.
C
      CALL UMOVE (0.,0.)
      CALL USET ('LNULL')
      IF(INDEX.EQ.1) CALL USET ('LA')
C
C     DISPLAY A LINE THEN BRANCH TO ONE OF FOUR ROUTINES TO PRINT AN 'A'
C     AT THE END OF THE LINE SEGMENT.
C
      DO 4 K=1,4
      CALL UPEN ((25.*FLOAT(K)),(25.*FLOAT(K)))
      GO TO (4,1,2,3), INDEX
    1 CALL UAOUT ('A;')
      GO TO 4
    2 CALL UPRNT1 ('A;', 'TEXT')
      GO TO 4
    3 CALL UWRIT1 ('A;', 'TEXT')
    4 CONTINUE
C
C     DISPLAY THE NAME OF THE ROUTINE WHICH WAS USED AT BOTTOM RIGHT
C     CORNER OF WINDOW.
C
C     CALL UPRINT (75.,2.,ROUTINE(INDEX))
    5 CONTINUE
C
C     WRAP-UP GRAPHICS ACTIVITY AND TERMINATE THIS FORTRAN PROGRAM
C
      CALL UEND
      STOP
      END
```

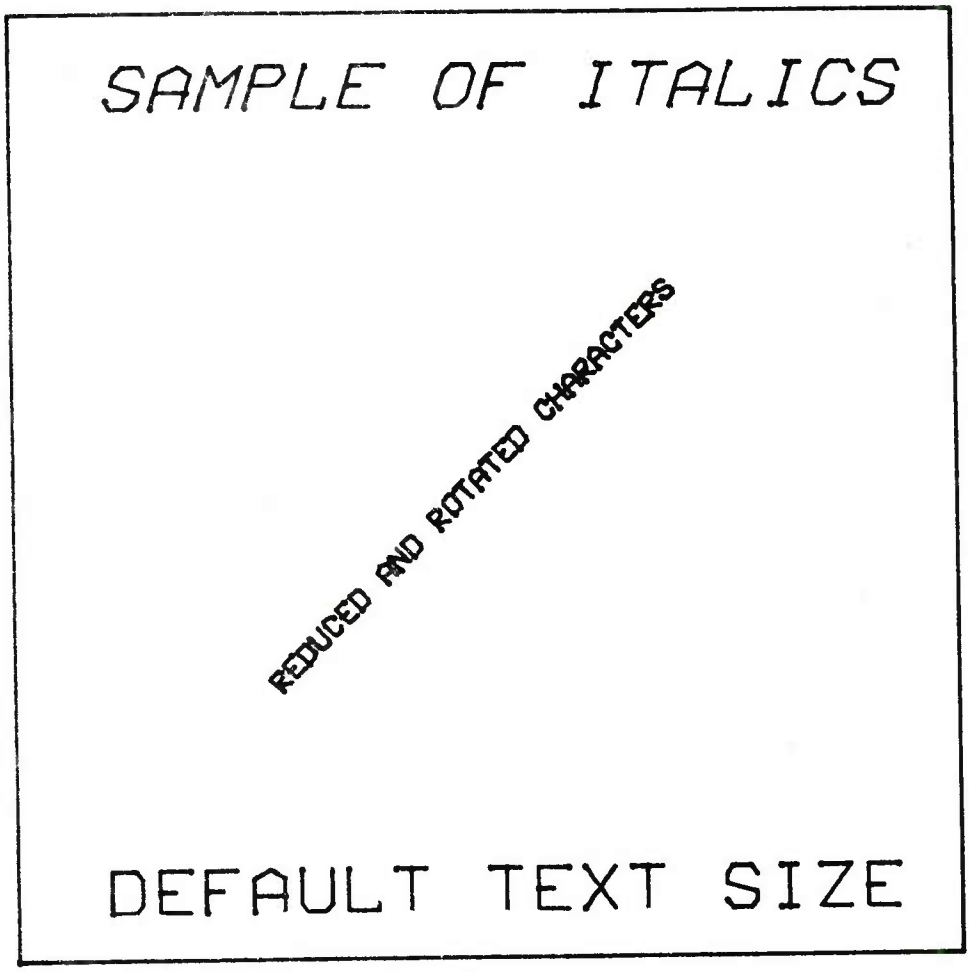
EXAMPLE IV-4




```
      CHARACTER ROUTINE*7(4)
      DATA INDEX,Y0/0,5.73/
      DATA ROUTINE/'UPEN;','UAOUT;','UPRNT1;','UWRIT1;'/
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      DO 5 I = 1, 2
      X0 = -1.82
      Y0 = Y0 - 2.86
      DO 5 J = 1, 2
      X0 = X0 + 2.86
      INDEX = INDEX + 1
      CALL UDAREA (X0,(X0+2.57),Y0,(Y0+2.57))
      CALL UOUTLN
      CALL UMOVE (0.,0.)
      CALL USET ('LNULL')
      IF (INDEX .EQ. 1) CALL USET ('LA')
      DO 4 K = 1, 4
      CALL UPEN ((25.*FLOAT(K)),(25.*FLOAT(K)))
      GO TO (4,1,2,3), INDEX
    1 CALL UAOUT ('A;')
      GO TO 4
    2 CALL UPRNT1 ('A;','TEXT')
      GO TO 4
    3 CALL UWRIT1 ('A;','TEXT')
    4 CONTINUE
      CALL UPRINT (75.,2.,ROUTINE(INDEX))
    5 CONTINUE
      CALL UEND
      STOP
      END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE ALPHANUMERIC OUTPUT VIA THE
C     'SOFTWARE' CHARACTER OPTION AVAILABLE UNDER GCS.  THREE
C     EXAMPLES ARE INCLUDED:  A/N OUTPUT USING THE DEFAULT SIZES OF
C     SOFTWARE CHARACTERS, A SAMPLE OF ITALICIZED OUTPUT, AND A
C     DEMONSTRATION OF REDUCED AND ROTATED A/N OUTPUT.
C
C     ENTER GCS, OUTLINE DEFAULT DEVICE PLOTTING AREA & SPECIFY THAT
C     SOFTWARE CHARACTERS ARE TO BE USED
C
      CALL USTART
      CALL UOUTLN
      CALL USET ('SOFTWARE')
      CALL UPSET ('TERMINATOR', ';')
C
C     OUTPUT SOME TEXT WHICH STARTS AT LOCATION (10,5) AND USES THE
C     DEFAULT SOFTWARE CHARACTER SIZE.
C
      CALL UPRINT (10.,5.,'DEFAULT TEXT SIZE;')
C
C     SPECIFY THAT ITALICIZED CHARACTERS ARE DESIRED, AND OUTPUT A
C     STRING WHICH BEGINS AT LOCATION (10,90).
C
      CALL USET ('ITALICS')
      CALL UPRINT (10.,90.,'SAMPLE OF ITALICS;')
C
C     CHANGE THE CHARACTER TYPE BACK TO GOTHIC, AND SPECIFY THE SIZE
C     OF THE NEW SOFTWARE CHARACTER DESIRED — 2 VIRTUAL UNITS BY 3
C     VIRTUAL UNITS.
C
      CALL USET ('GOTHIC')
      CALL UPSET ('HORIZONTAL',2.)
      CALL UPSET ('VERTICAL',3.)
C
C     MOVE TO THE ORIGIN OF THE CURRENT COORDINATE SYSTEM, AND
C     PERFORM A 45 DEGREE ROTATION ABOUT THIS ORIGIN.
C
      CALL UMOVE (0.,0.)
      CALL UROTAT (45.)
C
C     OUTPUT A STRING WHICH BEGINS AT LOCATION (40,0) WITHIN THE
C     ROTATED COORDINATE SYSTEM.
C
      CALL UPRINT (40.,0.,'REDUCED AND ROTATED CHARACTERS;')
C
C     WRAP-UP GRAPHICS ACTIVITY AND TERMINATE THE FORTRAN PROGRAM.
C
      CALL UEND
      STOP
      END
```

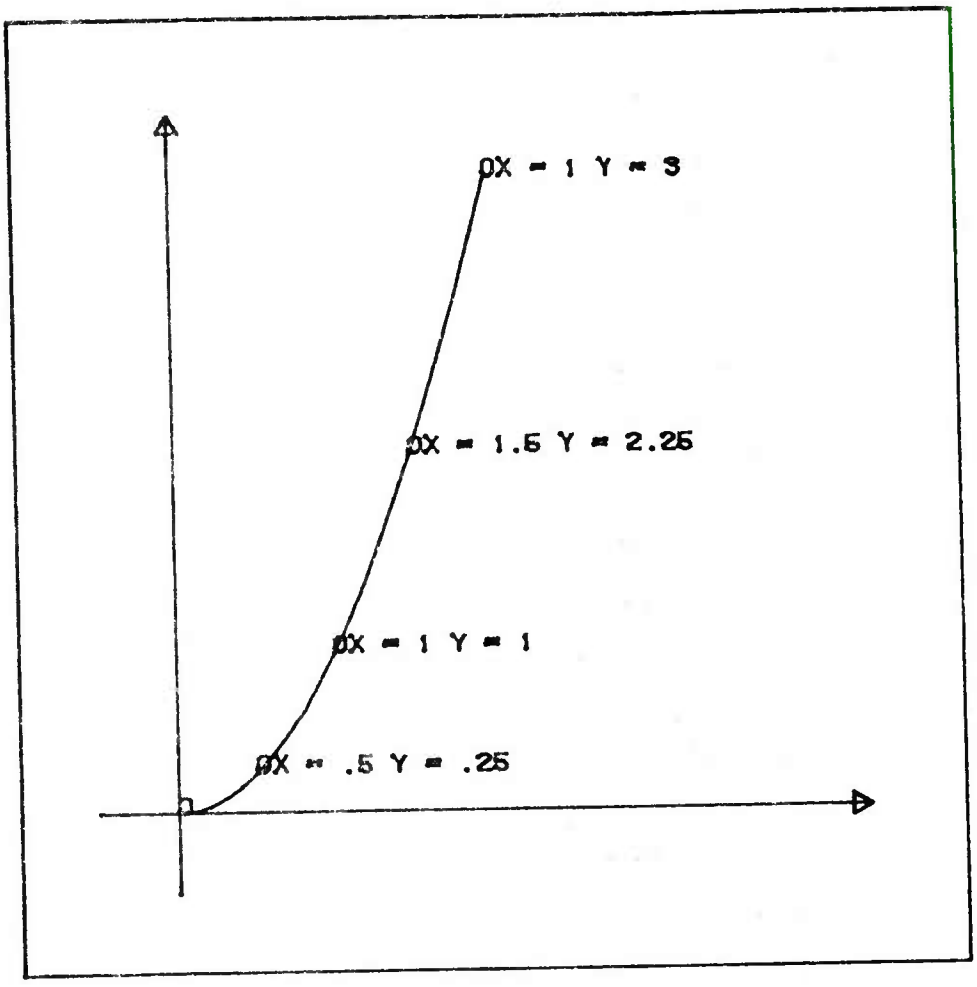
EXAMPLE IV-5




```
CALL USTART
CALL UOUTLN
CALL UPSET ('TERMINATOR',';')
CALL USET ('SOFTWARE')
CALL UPRINT (10.,5.,'DEFAULT TEXT SIZE;')
CALL USET ('ITALICS')
CALL UPRINT (10.,90.,'SAMPLE OF ITALICS;')
CALL USET ('GOTHIC')
CALL UPSET ('HORIZONTAL',2.)
CALL UPSET ('VERTICAL',3.)
CALL UMOVE (0.,0.)
CALL UROTAT (45.)
CALL UPRINT (40.,0.,'REDUCED AND ROTATED CHARACTERS;')
CALL UEND
STOP
END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE 'UPRINT' TO OUTPUT MIXED
C       ALPHANUMERIC DATA
C
C       INITIALIZE ALL VARIABLES WHICH WILL BE USED
C
        CHARACTER OPTION*10(4)/'REALNUMBER', 'INTEGER', 'REALNUMBER',
        'INTEGER'/
        CHARACTER TITLE*6(2)/'X = ; ', 'Y = ; '/
        DATA X,Y,/0.,0./
C
C       ENTER GCS, SET UP WINDOW AND DRAW AXES
C
        CALL USTART
        CALL UOUTLN
        CALL UWINDO (-1.,5.,-1.,5.)
        CALL UMOVE (0.,-.5)
        CALL UPEN1 (0.,4.4.,'LARROW')
        CALL UMOVE (-.5,0)
        CALL UPEN (4.4,0.,'LARROW')
C
C       MOVE TO ORIGIN AND MARK IT
C
        CALL UPEN (X,Y,'NO')
C
C       DRAW A SEGMENT OF A PARABOLA, MARK THE COORDINATES AT
C       INTERVALS OF .5 FOR X AND ALTERNATELY OUTPUT THE VALUES OF X AND
C       Y AS REAL AND INTEGER.
C       WHEN EXECUTING THIS PROGRAM ON CDC MACHINES, USE THE ROUND
        OPTION ON THE FTM COMPUTER IN ORDER TO ASSURE CORRECT
C       PRINTOUTS
C
        DO 1 I=1,4
        DO 2 J=1,5
        X=X+.10
        Y=X**2
        CALL UPEN (X,Y)
    2   CONTINUE
        CALL UPEN1 (X,Y,'NO')
        CALL UPRNT1 (TITLE(1),'TEXT')
        CALL UPRNT1 (X,OPTION(1))
        CALL UPRNT1 (TITLE(2),'TEXT')
        CALL UPRNT1 (Y,OPTION(1))
        CALL UMOVE (X,Y)
    1   CONTINUE
C
C       WRAP-UP GRAPHICS ACTIVITY AND TERMINATE PROGRAM.
C
        CALL UEND
        STOP
        END
```

EXAMPLE IV-6




```
      CHARACTER OPTION*10(4)
      DATA OPTION/'REALNUMBER','INTEGER',
     &            'REALNUMBER','INTEGER'/
      CHARACTER TITLE*8(2)/' X = ',' Y = '/
      DATA X,Y/0.,0./
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL UOUTLN
      CALL UWINDO (-1.,5.,-1.,5.)
      CALL UMOVE (0.,-.5)
      CALL UPEN1 (0.,4.4,'LARROW')
      CALL UMOVE (-.5,0.)
      CALL UPEN1 (4.4,0.,'LARROW')
      CALL UPEN1 (X,Y,'NO')
      DO 1 I = 1, 4
      DO 2 J = 1, 5
      X = X + .10
      Y = X**2
      CALL UPEN (X,Y)
    2 CONTINUE
      CALL UPEN1 (X,Y,'NO')
      CALL UPRNT1 (TITLE(1),'TEXT')
      CALL UPRNT1 (X,OPTION(I))
      CALL UPRNT1 (TITLE(2),'TEXT')
      CALL UPRNT1 (Y,OPTION(I))
      CALL UMOVE (X,Y)
    1 CONTINUE
      CALL UEND
      STOP
      END
```

CHAPTER V

GRAPHICAL AND ALPHANUMERIC INPUT

The following section discusses the facilities within GCS that are available for entering data of a graphical nature. Not all implementations of GCS can be used for such graphics input. For example, those implementations of GCS running in a batch mode, i.e., passive graphics, do not permit graphical input. For these applications, the routines discussed below are disabled.

For a graphics system to be truly interactive, it is necessary to provide the user with the ability of communicating information of a strictly graphical nature to his graphics program. Graphical input differs from conventional FORTRAN input (i.e., READ, etc.) in the following areas:

A. Graphical input is normally performed as a composite operation involving the transfer of positional coordinate information from the terminal's primary graphic input device (cursor, light pen, etc.) in conjunction with a single character from the keyboard (or other suitable interrupt) which signals the termination of the input operation.

B. Because of the unique nature of strictly graphical input, standard FORTRAN input cannot be used in lieu of software which has been specifically designed for that given purpose.

C. Graphical input does not produce any extraneous output at the graphics terminal when entering information, in contrast with standard FORTRAN free-formatted input which alerts the user through the use of a special symbol that input is requested, and echoes the input information as it is entered.

Positional and Character Graphical Input

The standard positional and character graphics input subroutine implemented within GCS is UGRIN, which may be called by the following sequence:

CALL UGRIN (X,Y,CHAR)

When UGRIN is invoked, the terminal's primary graphics input device is activated, and remains in an enabled state until a character is entered from the keyboard. Upon receipt of the character, UGRIN stores it into CHAR using standard FORTRAN Hollerith format, disables the primary input device, and transfers the positional coordinates (in current units) of the input device at the time the character was entered into X and Y. See Example V-1.

Subroutine UAIN functions in the same manner as UGRIN, except that no positional information is returned. Since the terminal remains in 'limbo' while awaiting input from the user, UAIN provides a convenient method of pausing at desired points during execution of a graphics program.

CALL UAIN (CHAR)

Alphanumeric Input (Fortran)

For bulk alphanumeric input used in conjunction with interactive communications and

coordination activities, standard FORTRAN may be utilized under GCS using the same criteria outlined in Chapter IV for FORTRAN alphanumeric output; i.e., the terminal must be placed into the alphanumeric mode through the call to UALPHA before control is relinquished to the FORTRAN input routines. It should be emphasized, however, that although FORTRAN input is extremely convenient, the use of FORTRAN input procedures significantly reduces the efficiency at which a graphics program executes; hence, the use of FORTRAN I/O should be avoided whenever possible.

Alphanumeric Input (Graphical)

An alternative method of alphanumeric input is provided to the graphics programmer through subroutine UREAD:

CALL UREAD (X,Y,DATA,COUNT,FLAG)

X and Y are used to specify the starting position (in current units) of where the alphanumeric input is to be attempted. UREAD will store the edited input into DATA, based upon one of the four options available to the user through a call of USET prior to invoking UREAD.

A. **TEXT** — Under this default option, UREAD will accept and store COUNT characters into DATA: hence, the user must insure that DATA has been suitably dimensioned to hold the number of characters which have been requested. Should fewer than COUNT characters be entered as input, UREAD will store the actual number of characters entered into FLAG, and blank-fill the remaining (COUNT - FLAG) characters of DATA. It should be noted that UREAD does not append the termination character to the end of the input. Therefore, the user is responsible for inserting this character via UAPEND if the string is to be passed as a parameter to UPRINT or UWRITE.

B. **REALNUMBER** — This option directs UREAD to edit the alphanumeric input as a REAL number, and to store the resulting floating-point number into the single-valued REAL parameter DATA. Should UREAD encounter any illegal characters during the edit, FLAG will be returned with a negative value, and DATA will be undefined.

C. **INTEGER** — As in the case of REALNUMBER, DATA is assumed to be single-valued and of type REAL. UREAD will edit the alphanumeric input as an INTEGER, perform an INTEGER to REAL conversion, and store the result into DATA. The user may check if the operation was successfully performed by examining FLAG upon return from UREAD.

D. **XYCOORDINATES** — This option directs UREAD to accept two REAL numbers (separated by a comma) as input and to store them into the two element REAL array, DATA. FLAG is set to reflect the status of the input and editing operation.

Under **REALNUMBER, INTEGER, and XYCOORDINATES** options, COUNT is used to specify the number of variables which are to be input under that given mode. For example, if two integers were desired to be read into DATA at virtual location (0.100.), the following call to UREAD could be used:

```
CALL USET ('INTEGER')
CALL UREAD (0.,100.,DATA,2.,FLAG)
```

Under **XYCOORDINATES,** COUNT designates the number of ordered pairs to be input.

A streamlined version of UREAD is available to the user under GCS through subroutine UINPUT:

CALL UINPUT (DATA,COUNT,FLAG,OPTION)

DATA, COUNT and FLAG are interpreted in the same manner as UREAD; OPTION is a single-valued character variable which specifies the format or mode under which the input is to be editted and stored into DATA, i.e., TEXT, REALNUMBER, INTEGER, etc. The UINPUT/UREAD relationship closely parallels UPRNT1/UPRINT in that the operation is performed at the current beam position, and the effects of OPTION are purged from the GCS upon exit from the subroutine. See Example V-2.

Menued Graphical Input

Aside from entering positional information which directs the generation of a display, the bulk of most interactive input requirements centers upon the ability to select one of several options which are 'menued' before the user. Menuing represents a very attractive means of user interaction due to the inherent simplicity of the required user response -- one need only position the graphic input device within the square of the menuboard which represents the desired option, and depress a character on the keyboard to transmit the selection to the user's program.

Within GCS, graphical input, by means of a menuboard, is facilitated through the use of subroutine UMENU, which is called by the following sequence:

CALL UMENU (PTSIN,LABELS,CHOICE)

The absolute value of PTSIN represents the number of menu options that will be provided (maximum of 10); LABELS is a character array containing one entry of up to eight characters which is to be printed under each menu option; and CHOICE is a single-valued output REAL variable, assigned a value by UMENU which indicates the box number of the option which the user has selected. When UMENU is called with a positive value of PTSIN, a menuboard is drawn, and the graphic input device is enabled. A negative value of PTSIN merely enables the input device and inhibits the redrawing of the menuboard. See Example V-3.

Drafting-Type Graphical Input

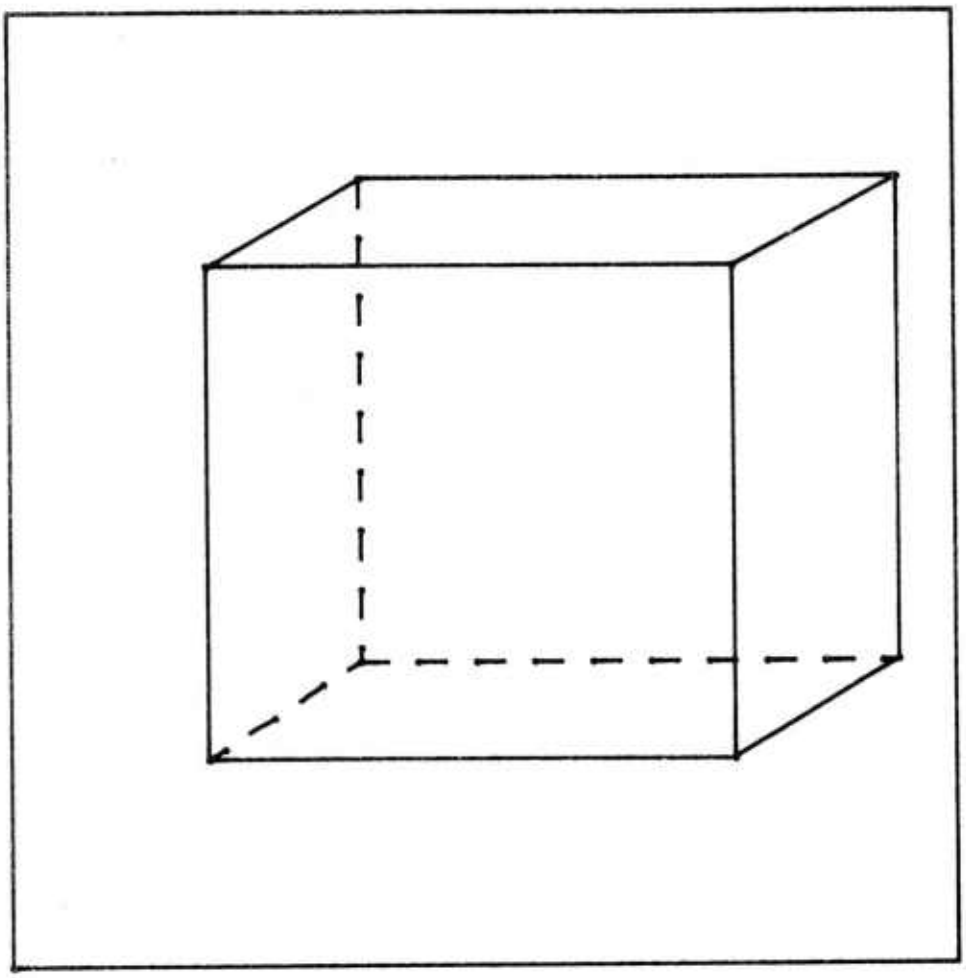
The three GCS graphical input subroutines described in this chapter represent a core around which many sophisticated interactive graphical programs may be designed. There exists one additional GCS graphical input subroutine, UDRIN, which permits high-level drafting-type activities to be performed under program control; in addition, the result of the man/machine interaction may be saved and recalled during future interactions.

CALL UDRIN(X,Y,CHAR)

Interested users are directed to the GCS Programmer's Reference Manual for further discussion.

```
C
C       SAMPLE PROGRAM WHICH ILLUSTRATES GRAPHICS INPUT APPLICATION
C       THROUGH 'UGRIN'. THREE TYPES OF CASES ARE HANDLED: SOLID 'S'
C       INVISIBLE 'I', AND DASHED 'D' LINES. THE DESIRED OPTION FOR THE LINE IS
C       ENTERED AS A SINGLE CHARACTER WHEN THE CURSORS HAVE BEEN
C       POSITIONED. DEFAULT VALUES OF WINDOW, DEVICE AREA, AND DASH
        SPECIFICATION ARE USED. AN 'E' WILL TERMINATE THE PROGRAM.
C
        CHARACTER CHAR*1
        CALL USTART
        CALL UOUTLN
C
C       ENABLES CURSORS, OBTAIN (X,Y) COORDINATES, AND THE CHARACTER.
C
    1   CALL UGRIN (X,Y,CHAR)
C
C       CHECK FOR A REQUEST FOR A SOLID LINE TO BE DRAWN TO (X,Y).
C
        IF (CHAR.EQ.'S') CALL UPEN1 (X,Y,'LINE')
C
C       CHECK FOR A REQUEST FOR AN INVISIBLE LINE (MOVE) TO (X,Y).
C
        IF (CHAR.EQ.'I') CALL UMOVE (X,Y)
C
C       CHECK FOR A REQUEST TO DRAW A DASHED LINE TO POINT (X,Y).
C
        IF (CHAR.EQ.'D') CALL UPEN1 (X,Y,'DASH')
C
C       CHECK IF THE USER DESIRES TO TERMINATE CURRENT GCS PROGRAM.
C
        IF (CHAR.EQ.'E') GO TO 2
        GO TO 1
    2   CALL UEND
        STOP
        END
```

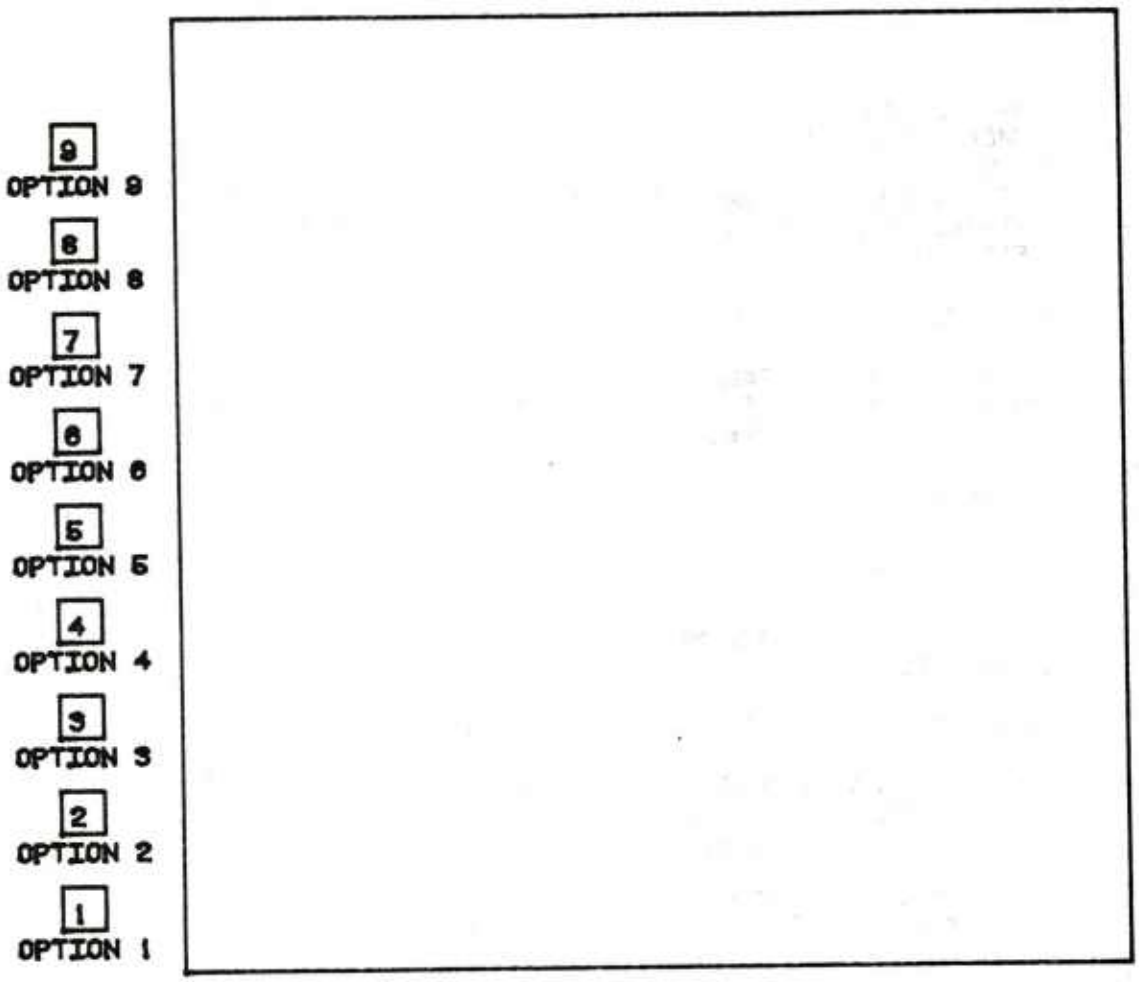
EXAMPLE V-1



```
      CHARACTER CHAR*1
      CALL USTART
      CALL UOUTLN
    1 CALL UGRIN (X,Y,CHAR)
      IF (CHAR .EQ. 'S') CALL UPEN1 (X,Y,'LINE')
      IF (CHAR .EQ. 'I') CALL UMOVE (X,Y)
      IF (CHAR .EQ. 'D') CALL UPEN1 (X, Y, 'DASH')
      IF (CHAR .EQ. 'E') GO TO 2
      GO TO 1
    2 CALL UEND
      STOP
      END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF SUBROUTINE
C     'UMENU'. THE CHARACTER ARRAY 'OPTION' CONTAINS THE LABELS TO BE
C     PRINTED UNDER EACH OF THE MENU SELECTION BOXES. THE NUMBER OF
C     THE BOX WHICH WAS SELECTED BY THE USER IS RETURNED IN THE
C     PARAMETER 'CHOICE'.  THE DEFAULT VALUES OF VIRTUAL WINDOW AND
C     DEVICE AREA ARE USED.
C
C     INITIALIZE LABELS FOR THE MENU CHOICES
C
      CHARACTER OPTION*8(9)
      DATA OPTION/'OPTION 1', 'OPTION 2', 'OPTION 3', 'OPTION 4', 'OPTION 5',
                  'OPTION 6', 'OPTION 7', 'OPTION 8', 'OPTION 9'/
C
C     ENTER GCS
C
      CALL USTART
      CALL UOUTLN
C
C     CALL 'UMENU' TO DRAW THE MENUBOARD OF 9 OPTIONS AND ACCEPT THE
C     USER'S SELECTION
C
      CALL UMENU (9.0, OPTION, CHOICE)
C
C     CALL 'UMENU' ONCE AGAIN, BUT USE A MINUS SIGN (-) TO SPECIFY THAT
C     THE MENUBOARD IS NOT TO BE REDRAWN, BUT THAT THE USER IS TO INPUT
C     ANOTHER CHOICE (OR REENTER AN EARLIER ONE)
C
      CALL UMENU (-9.0, OPTION, CHOICE)
      CALL UEND
      STOP
      END
```

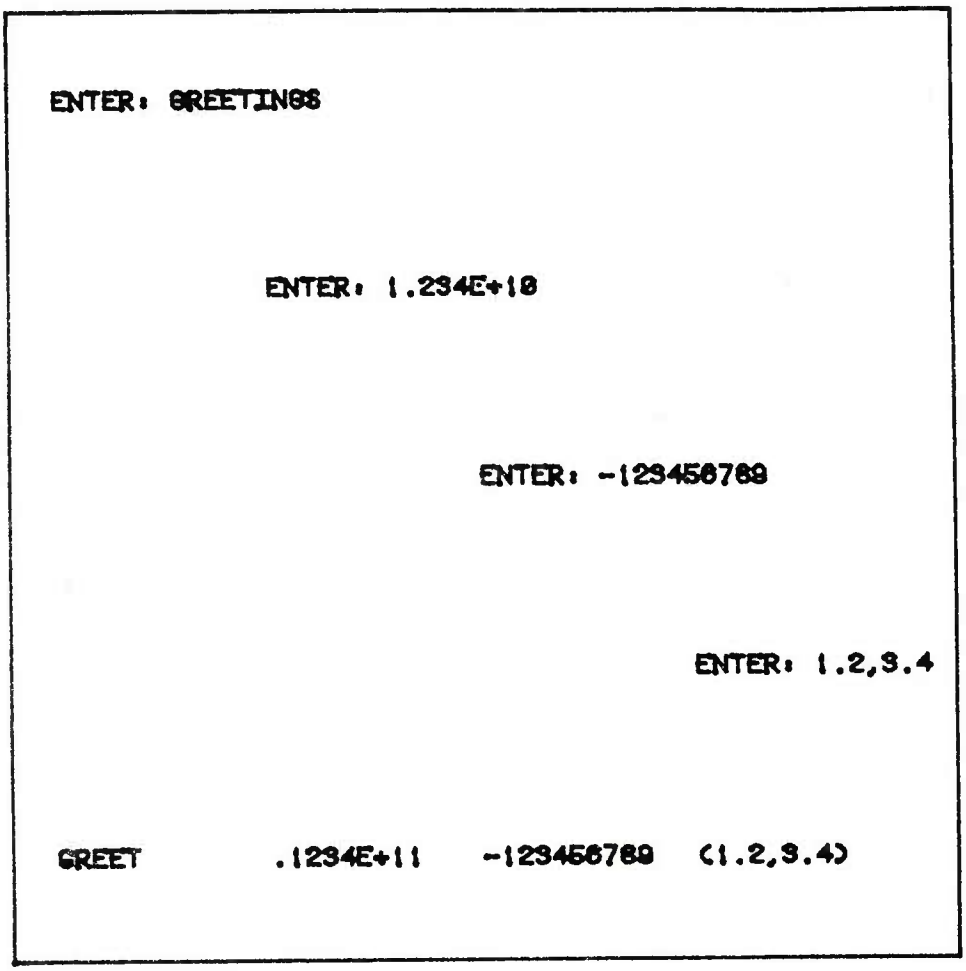
EXAMPLE V-2




```
      CHARACTER OPTION*8(9)
      DATA OPTION/'OPTION 1','OPTION 2','OPTION 3','OPTION 4',
     &            'OPTION 5','OPTION 6','OPTION 7','OPTION 8',
     &            'OPTION 9'/
      CALL USTART
      CALL UOUTLN
      CALL UMENU (9.0,OPTION,CHOICE)
      CALL UMENU (-9.0,OPTION,CHOICE)
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF SUBROUTINE
C       'UINPUT' TO ACCEPT ALPHANUMERIC INPUT FROM A USER, EDIT IT INTO THE
C       PROPER FORMAT, STORE IT INTO A DATA ARRAY, & PRINT THE DATA ARRAY
C       AT A DIFFERENT LOCATION ON THE DISPLAN.
C
C       DEFINE AND INITIALIZE DATA ARRAYS USED IN THIS GCS EXAMPLE
C
        CHARACTER OPTION*12(4)
        DIMENSION COUNT(4),DATA(6),INDEX(4)
        DATA COUNT, INDEX,X,Y/5.,1.,1.,1.,1,3,4,5,5.,90./
        DATA OPTION/'TEXT','REALNUMBER','INTEGER','XYCOORDINATE'/
C
C       ENTER GCS, OUTLINE DEFAULT VIRTUAL WINDOW, AND DEFINE LOOP TO
C       ILLUSTRATE THE FOUR INPUT AND OUTPUT OPTIONS.
C
        CALL USTART
        CALL UPSET ('TERMINATOR',';')
        CALL UOUTLN
        DO 1 I=1,4
C
C       POSITION BEAM/PEN TO PROPER LOCATION PRIOR TO PRINTING OF A
C       PROMPTING MESSAGE.
C
        CALL UMOVE (X,Y)
C
C       ALERT THE USER THAT INPUT IS DESIRED, THEN ACCEPT THE DATA
C
        CALL UPRNT1 ('ENTER: ; 'TEXT')
        CALL UINPUT (DATA(INDEX(1)), COUNT(2), FLAG, OPTION(2))
C
C
C       OUTPUT DATA
        IF(1.EQ.1) CALL UAPEND (COUNT(I), DATA (1)), DATA(INDEX(1))
        CALL USET (OPTION(I))
        CALL UPRINT (X,10.,DATA(INDEX(I)))
C
C       UPDATE COORDINATE LOCATIONS FOR NEXT ATTEMPT AT A/N INPUT
C
        X=X+22.5
        Y=Y-20.
    1   CONTINUE
C
C       WRAP-UP GRAPHICS ACTIVITY, & TERMINATE THE FORTRAN PROGRAM.
C
        CALL UEND
        STOP
        END
```

EXAMPLE V-3




```
      CHARACTER OPTION*12(4)
      DIMENSION COUNT(4),DATA(6),INDEX(4)
      DATA COUNT,INDEX,X,Y/5.,1.,1.,1.,1,3,4,5,5.,90./
      DATA OPTION/'TEXT','REALNUMBER','INTEGER','XYCOORDINATE'/
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL UOUTLN
      DO 1 I = 1, 4
      CALL UMOVE (X,Y)
      CALL UPRNTI ('ENTER: ;','TEXT')
      CALL UINPUT (DATA(INDEX(I)),COUNT(I),FLAG,OPTION(I))
      IF(I.EQ.1) CALL UAPEND(COUNT(1),DATA(INDEX(1)),DATA(INDEX(1)))
      CALL USET (OPTION(I))
      CALL UPRINT (X,10.,DATA(INDEX(I)))
      X = X + 22.5
      Y = Y - 20.
    1 CONTINUE
      CALL UEND
      STOP
      END
```

CHAPTER VI

GCS UTILITY SUBROUTINES

To assist the user in preparing his graphical applications, GCS has incorporated a series of comprehensive, yet easy to use utility subroutines which permit the programmer to generate:

A. Circles and circular arcs

B. Regular polygons and other straight-sided geometric figures

C. Conic sections

D. Linear and least-squares polynominal curve fits to a series of user-supplied data points.

All of the GCS utility routines which generate graphical output perform their activities within the environment established by the user at the time they were invoked; hence, such output will be constrained by such parameters as the virtual window and device area, current units and mode of addressing, line type and terminator, together with any rotational and scaling factors specified by the user.

Circles

One of the most common requirements of scientific and engineering graphical applications is fo the generation of circles and circular arcs. Provision for this capability under GCS is available through subroutines UCRCLE and UARC.

Circles may be generated through subroutine UCRCLE which is called by the following sequence:

CALL UCRCLE (X,Y,RADIUS)

X and Y are used to specify the coordinates (in current units) of the center of the circle of radius, RADIUS, which will be approximated by a series of line segments forming a many-sided equilateral polygon, whose peak deviation from an ideal circle is approximately 1.4 thousandths of the ideal circle's radius. This error is normally comparable in magnitude to the raster spacing of the plotting device, and should prove adequate under most conditions. Example VI-1 demonstrates the use of UCRCLE.

Circular Arcs

Circular arcs may be generated through subroutine UARC which is called by the following sequence:

CALL UARC (X,Y,ANGLE)

Upon invocation, UARC will generate an arc beginning at the current beam position and of angular span ANGLE. The radius of the arc is determined through computation of the distance from the center of the arc (specified by X and Y) and the beam position prior to entry into the subroutine. Upon exit, UARC will leave the beam at the last point of the angular segment it has generated. For example, if all default conditions are in effect, the current beam position is at (40.0,50.0), and UARC is invoked with the following parameters:

CALL UARC (40.0,20.0,90.0)

UARC will generate a circular arc having a center at (40.0,20.0), with a radius of 30.0, and angular span of 90.0 degrees. The arc will begin at (40.0,50.0) and terminate at (10.0,20.0). See Example VI-2.

Should the user be at a terminal which has the capability to generate circles and circular arcs via hardware, UCRCLE and UARC will utilize such facilities when they are invoked.

Regular Polygons

Polygon and other straight-sided geometric figure generation is provided through subroutines UPLYGN and URECT. Subroutine URECT may be utilized when a rectangle, which spans the area defined by the current beam position and coordinates (X,Y), is desired. Should URECT be called under the RELATIVE mode of addressing, any rotational or scaling parameters which may apply will dictate the rotation or scaling of the rectangle about the current beam position.

CALL RECT(X,Y)

Provision for the generation of equilateral (regular) polygons is available under GCS through UPLYGN which may be called by the following sequence:

CALL UPLYGN (X,Y,PTS,RADIUS)

X and Y denote the coordinates of the center of the polygon, PTS is the number of sides which is to comprise the figure, and RADIUS is used to define the radius of the circle within which the polygon is inscribed. With no relative rotation specified, the polygon will be drawn with one side parallel to the bottom of the plotting area. Should relative rotation be specified, this "base" side will be rotated by the given angle of rotation. In contrast to URECT, UPLYGN's axis of rotation is defined by (X,Y) as opposed to the beam position upon entry to the subroutine. Examples VI-3 through VI-6 demonstrate the use of UPLYGN and URECT and illustrates some of the principles of rotation and line options.

Conic Sections

A particularly useful subroutine which may be used to draw all (or part) of a generalized conic section having a given focus, directrix, eccentricity, and angular span is available through the following calling sequence:

CALL UCONIC (X,Y,P,E,THETA1,THETA2)

(X,Y) are used to specify the coordinates of the focus of the conic section; P is the distance from the focus to the directrix; E is the eccentricity, THETA1 and THETA2 represent the initial and final angles through which the conic section is to be drawn. Parameters E and P affect the generation of the conic section in the following manner:

A.	E=0	will draw a circular arc with a center at (X,Y) and radius P/2. The arc will subtend the angular range defined by THETA1 and THETA2.
B.	0.LT.ABS(E).LT.1	will generate an ellipse.
C.	ABS(E)=1	will specify a parabola.
D.	ABS(E).GT.1	designates that a hyperbola is to be drawn.

E.	E.GT.0	indicates that the major axis of the conic section is to be oriented parallel to the X-axis.
F.	E.LT.0	specifies that the major axis is to be oriented along the Y-axis.
G.	P.GT.0	defines the position of the focus to be to the right of (or below) the directrix.
H.	P.LT.0	indicates that the focus is positioned to the left of (or above) the directrix.

Examples VI-7 and VI-8 demonstrate the use of UCONIC.

Curve Fitting

For those users who desire to combine data analysis through curve fitting into their graphical applications, subroutines ULINFT and ULSTSQ should provde of particular importance. ULINFT should be used when it is desired to obtain the slope (S) and Y-intercept (YI) of the linear equation, Y = SX + YI, which represents the least-squares linear fit to a number (XN) of user-supplied data points (contained in arrays X and Y). ULINFT may be called by the following sequence:

CALL ULINFT (X,Y,SN,S,YI)

See Example VI-9.

Subroutine ULSTSQ should be used when a 'least-squares' polynominal curve fit is desired. ULSTSQ returns the N+1 coefficients of the polynominal of degree N which represents the 'best' in the sense of least-square fit to a series of user-supplied data points. ULSTSQ may be invoked by:

CALL ULSTSQ (X,Y,XN,COEFF)

where X is a user-supplied array containing XN values of the independent variable; Y is an array of XN values for the dependent variable; XN is the number of values in each of the X and Y arrays; and COEFF is an output array which contains the N+1 coefficients of the polynominal which was fitted to the data. In order to specify the degree of the polynominal which is to be fitted to the data, it is necessary to call UPSET before ULSTSQ is invoked. For the generalized case outlined above, a suitable call to UPSET would be:

CALL UPSET ('POLYNOMINAL',FLOAT(N))

See Example VI-10.

Development of Applications Library

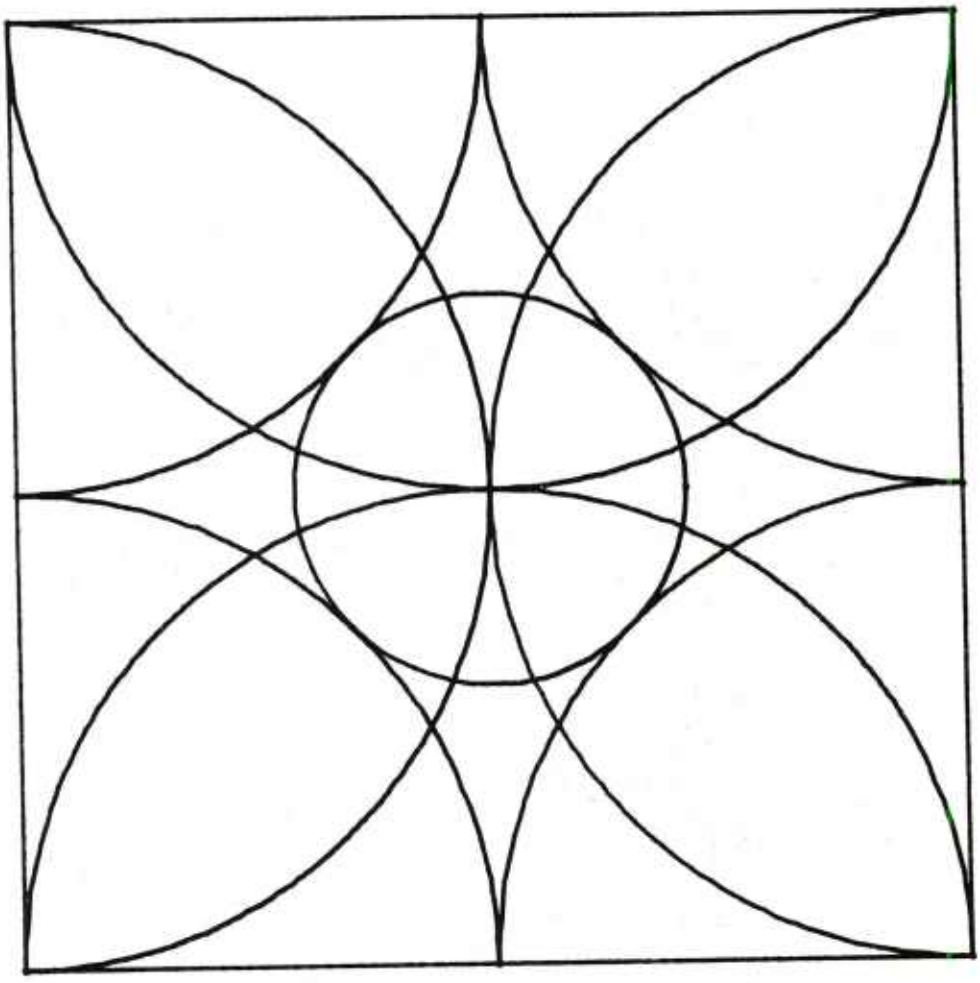
There is one additional curve fitting routine available, USPLIN. This routine does a cubic spline fit to a series of data points. The quality of fit is established by the nature of the cubic spline function. The user is referred to the GCS programmer's reference manual.

CALL USPLIN(X,Y,XN,RX,RY,RN)

The subroutines discussed in this chapter represent a group of software which may be viewed as a minature applications library.

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE SIMPLE APPLICATION OF THE
C     SUBROUTINE 'UCRCLE'.  DEFAULT VIRTUAL WINDOW AND DEVICE AREA IS
C     USED.  NOTE THE EFFECT OF 'SCISSORING' DUE TO THE VIRTUAL WINDOW'S
C     RESTRICTING OF ALL GRAPHICAL INFORMATION TO RESIDE WITHIN THE
C     REGION DEFINED BY A 100.-BY-100. SQUARE.
C
      CALL USTART
      CALL UOUTLN
C
C     DRAW CIRCLES WITH CENTERS AT CORNER OF THE VIRTUAL WINDOW, AND
C     CENTERS AT THE MIDDLE OF EACH OF THE WINDOW BOUNDARIES.  ALL OF
C     THESE CIRCLES HAVE A RADIUS OF 50.
C
      CALL UCRCLE (0.,0.,50.)
      CALL UCRCLE (50.,0.,50.)
      CALL UCRCLE (100.,0.,50.)
      CALL UCRCLE (100.,50.,50.)
      CALL UCRCLE (100.,100.,50.)
      CALL UCRCLE (50.100.,50.)
      CALL UCRCLE (0.,100.,50.)
      CALL UCRCLE (0.,50.,50.)
C
C     DRAW A CIRCLE OF RADIUS 20.7 WITH A CENTER AT (50.,50.).
C
      CALL UCRCLE (50.,50.,20.7)
      CALL UEND
      STOP
      END
```

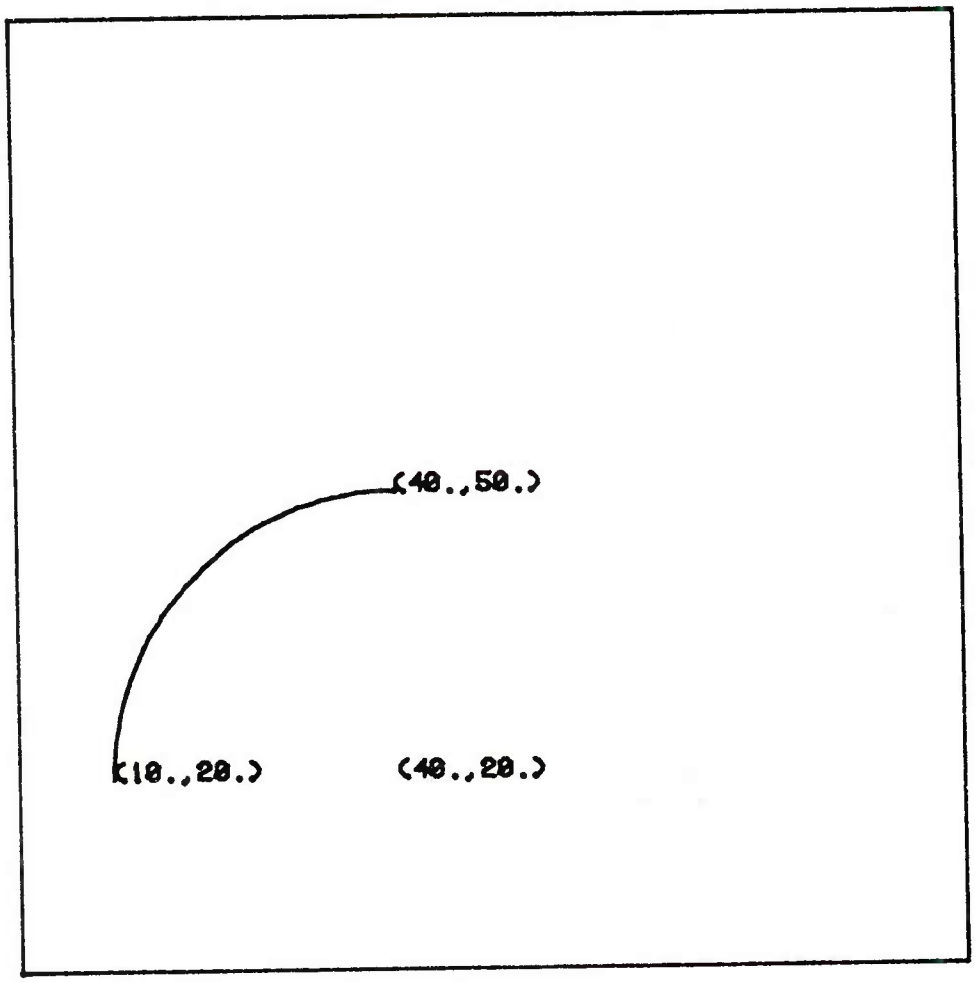
EXAMPLE VI-1



```
CALL USTART
CALL UOUTLN
CALL UCRCLE (0.,0.,50.)
CALL UCRCLE (50.,0.,50.)
CALL UCRCLE (100.,0.,50.)
CALL UCRCLE (100.,50.,50.)
CALL UCRCLE (100.,100.,50.)
CALL UCRCLE (50.,100.,50.)
CALL UCRCLE (0.,100.,50.)
CALL UCRCLE (0.,50.,50.)
CALL UCRCLE (50.,50.,20.7)
CALL UEND
STOP
END
```

```
C     SAMPLE PROGRAM USED TO DEMONSTRATE THE USE OF SUBROUTINE
C     'URAC' TO DRAW AN ARC CENTERED AT (40.,20.) AND OF RADIUS 30.0. THE
C     ARC WILL BEGIN AT (40.,50.), AND TERMINATE AT (10.,20.).
C
      CALL USTART
      CALL UOUTLN
C
C     MOVE TO WHERE THE ARC IS TO BEGIN AND LABEL THE POSITION.
C
      CALL UPEN1 (40.,0,50.0, 'NCOORDINATES')
      CALL UMOVE
C
C     CALL 'UARC' TO GENERATE AN ARC WHOSE CENTER IS (40,20.), SPANNING
      AN ANGULAR INTERVAL OF 90 DEGREES.
C
      CALL UARC (40.0,20.0,90.0)
C
C     DETERMINE WHERE 'UARC' HAS LEFT THE BEAM, THEN PRINT THE
C     COORDINATES OF THIS LOCATION. ALSO PRINT THE COORDINATE OF THE
C     CENTER OF THE ARC
C
      CALL UWHERE (X,Y)
      CALL UPEN1 (X,Y,'NCOORDINATES')
      CALL UPEN1 (40.0,20.0,'NCOORDINATES')
      CALL UEND
      STOP
      END
```

EXAMPLE VI-2

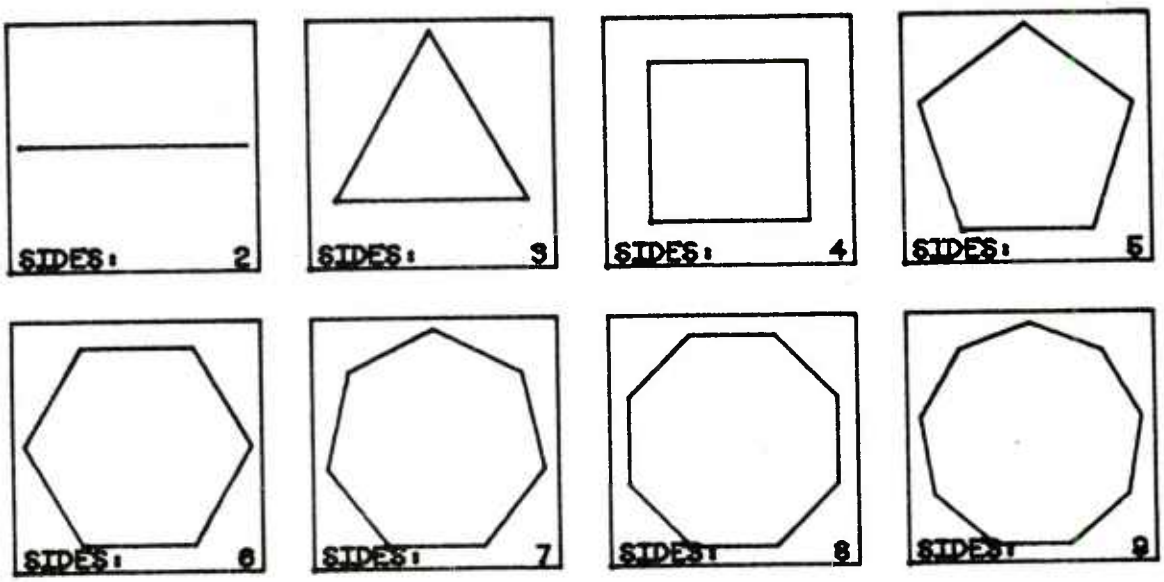



```
CALL USTART
CALL UOUTLN
CALL UPEN1 (40.0,50.0,'NCOORDINATES')
CALL UMOVE (40.0,50.0)
CALL UARC (40.0,20.0,90.0)
CALL UWHERE (X,Y)
CALL UPEN1 (X,Y,'NCOORDINATES')
CALL UPEN1 (40.0,20.0,'NCOORDINATES')
CALL UEND
STOP
END
```

```
C     SAMPLE PROGRAM USED TO DEMONSTRATE USE OF SUBROUTINE
C     'UPLYGN'. POLYGONS OF FROM 2 TO 9 SIDES WILL BE DRAWN WITHIN THEIR
C     OWN WINDOW, WHICH IS MAPPED TO DIFFERENT PORTIONS OF THE
C     SCREEN OF A TEKTRONIX 4010/4013 TERMINAL.
C
C     PRESET NUMBER OF SIDES OF POLYGON =1, 'INITIAL Y HEIGHT = 4.7'
C
      DATA SIDES YO/1.0,4.7/
C
C     START GCS. SET VIRTUAL WINDOW TO XMIN=-1.1 XMAX=+1.1 YMIN=-1.1
C     YMAX=+1.1. A DEVICE PLOTTING AREA WILL BE DEFINED 8 TIMES: 2 ROWS
C     OF 4, EACH TIME CONTAINING A DIFFERENT POLYGON AND WITH THE
C     DEVICE PLOTTING AREA OUTLINED.

      CALL USTART
      CALL UPSET ('TERMINATOR', ';')
      CALL UWINDO (-1.1,1.1,-1.1,1.1)
C
C     THE FOLLOWING DO-LOOPS SET UP THE 2 ROWS OF 4 DISPLAYS
C
      DO 1 I=1,2
      X=1.5
      YO=YO-1.8
      DO 1 J=1,4
      XO-XO+1.8
C
C     INITIALIZE NUMBER OF SIDES, 1 IS ADDED BEFORE EACH EXECUTION SO
C     POLYGON SIDES START AT 2 AND GO TO 9 IN 8 STEPS
C
      SIDES=SIDES+1.0
C
C     NOW ACTUALLY SET UP THE DEVICE PLOTTING AREA AS XO AND YO
C     CHANGE IT WILL MOVE TO 8 DIFFERENT LOCATIONS
C
      CALL UDAREA (XO,(XO+1.5),YO,(YO+1.5))
C
C     AT EACH LOCATION OUTLINE IT
C
      CALL UOUTLN
C
C     CALL 'UPLYGN' TO DRAW THE POLYGON WITHIN THE DEVICE AREA WE HAVE
C     DEFINED, THEN ADD SOME LABELING VIA CALLS TO 'UPRINT'.
C
      CALL UPLYGN (0.0,0.0,SIDES,1.0)
C
C     LABEL DRAWING
C
      CALL USET ('TEXT')
      CALL UPRINT (-1.0,-1.05,'SIDES;')
      CALL USET ('INTEGER')
      CALL UPRINT (0.9,-1.05,SIDES)
    1 CONTINUE
      CALL UEND
      STOP
      END
```

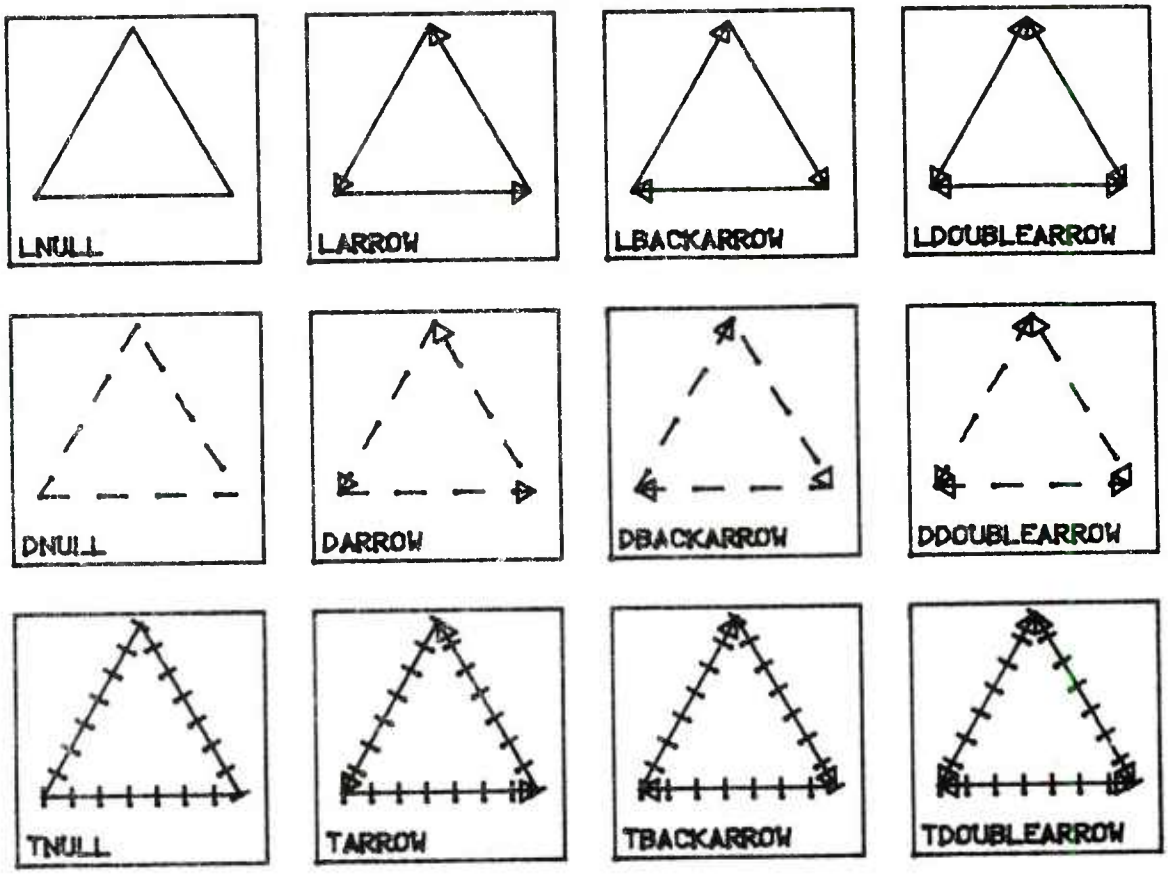
EXAMPLE VI-3




```
      DATA SIDES,Y0/1.0,4.7/
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL UWINDO (-1.1,1.1,-1.1,1.1)
      DO 1 I = 1, 2
      X0 = -1.5
      Y0 = Y0 - 1.6
      DO 1 J = 1, 4
      X0 = X0 + 1.6
      SIDES = SIDES + 1.0
      CALL UDAREA (X0,(X0+1.5),Y0,(Y0+1.5))
      CALL UOUTLN
      CALL UPLYGN (0.0,0.0,SIDES,1.0)
      CALL USET ('TEXT')
      CALL UPRINT (-1.0,-1.05,'SIDES:;')
      CALL USET ('INTEGER')
      CALL UPRINT (0.0,-1.05,SIDES)
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C     SAMPLE PROGRAM USED TO DEMONSTRATE USE OF SUBROUTINE
C     'UPLYGN'.  A TRIANGLE WILL BE DRAWN USING 1 OF 12 POSSIBLE PEN
C     OPTIONS ON A TETRONIX 4010/4013 TERMINAL.
C
C     LOAD A CHARACTER ARRAY WITH OPTIONS TO BE USED BY USET AND
      UPRINT
C
      CHARACTER OPTION*16(12)
      DATA INDEX,YO,OPTION/0.5,6,'LNULL;','LARROW:','LBACKARROW;'
     'LDOUBLEARROW;','DNULL;','DARROW;','DBACKARROW;',
     'DOUBLEARROW;','TNULL;','TARROW;','TBACKARROW;','TDOUBLARROW;'/
C
C     ENTER GCS, DEFINE VIRTUAL WINDOW.
C
      CALL USTART
      CALL UPSET ('TICINTERVAL', 0.25)
      CALL UPSET ('TERMINATOR',';')
      CALL UWINDO (-1.1,1.1,-1.1,1.1)
C
C     DRAW FIGURES IN THREE ROWS OF FOUR.
C
      DO 1 I=1,3
      XO=1.5
      YO=YO-1.8
      DO 1 J=1,4
      XO=XO+1.8
      INDEX=INDEX+1
C
C     DEFINE DEVICE AREA SO THAT ONLY ONE POLYGON WILL BE DRAWN.
C
      CALL UDAREA (XO,(XO+1.5),YO,(YO+1.5))
      CALL UOUTLN
C
C     CALL 'UPLYGN' TO DRAW THE POLYGON WITHIN THE DEVICE AREA WE HAVE
C     DEFINED USING ONE OF THE TWELVE POSSIBLE PEN OPTIONS.
C
      CALL USET (OPTION(INDEX))
      CALL UPLYGN (0.,0.0,3.0,1.0)
      CALL UPRINT (-1.0,-1.0,OPTION(INDEX))
    1 CONTINUE
      CALL UEND
      STOP
      END
```

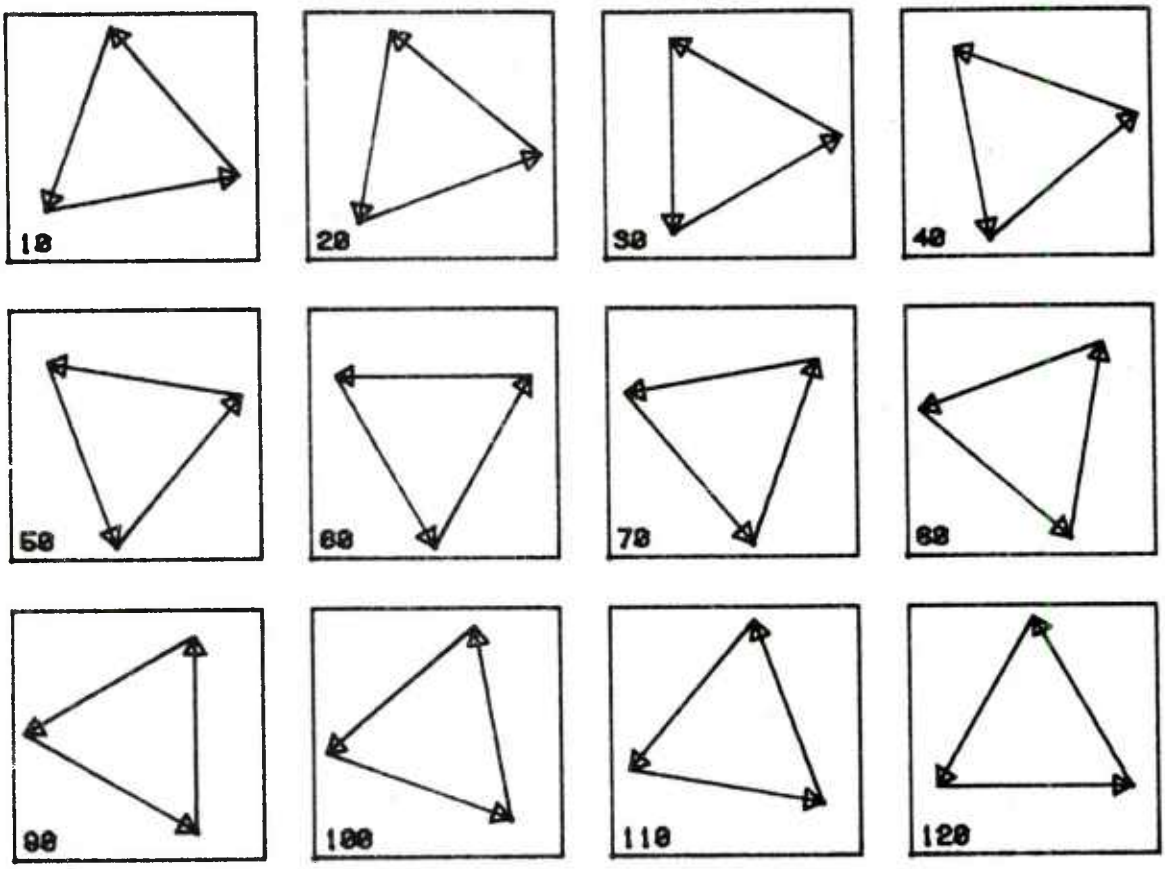
EXAMPLE VI-4




```
      CHARACTER OPTION*16(12)
      DATA INDEX,Y0/0,5.6/
      DATA OPTION/'LNULL;','LARROW;','LBACKARROW;','LDOUBLEARROW;',
     &            'DNULL;','DARROW;','DBACKARROW;','DDOUBLEARROW;',
     &            'TNULL;','TARROW;','TBACKARROW;','TDOUBLEARROW;'/
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL UPSET ('TICINTERVAL',0.25)
      CALL UWINDO (-1.1,1.1,-1.1,1.1)
      DO 1 I = 1, 3
      X0 = -1.5
      Y0 = Y0 - 1.8
      DO 1 J = 1, 4
      X0 = X0 + 1.8
      INDEX = INDEX + 1
      CALL UDAREA (X0,(X0+1.5),Y0,(Y0+1.5))
      CALL UOUTLN
      CALL USET (OPTION(INDEX))
      CALL UPLY0N (0.0,0.0,3.0,1.0)
      CALL UPRINT (-1.0,-1.0,OPTION(INDEX))
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO DEMONSTRATE USE OF SUBROUTINE
C       'UPLYGN'. A TRIANGLE WILL BE DRAWN IN RELATIVE MODE AND ROTATED
C       ABOUT ITS CENTER IN TEN DEGREE INCREMENTS ON A TEKTRONIX 4010/
C       4013 TERMINAL.
C
        DATA DEGREE,YO/0.0,5.563/
C
C       ENTER GCS, DEFINE VIRTUAL WINDOW, AND PEN OPTIONS
C
        CALL USTART
        CALL UWINDO (-1.1,1.1,-1.1,1.1)
        CALL USET ('INTEGER')
        CALL USET ('LARROW')
        DO 1 I=1,3
        XO=-1.5
        YO=YO-1.8
        DO 1 J=1,4
        XO=XO+1.8
        DEGREE=DEGREE+10.0
C
C       DEFINE DEVICE AREA SO THAT ONLY ONE POLYGON WILL BE DRAWN
C
        CALL UDAREA (XO,(XO+1.5),YO,(YO+1.5)
        CALL UOUTLN
C
C       CALL 'UPLYGN' TO DRAW THE POLYGON WITHIN THE DEVICE AREA WE HAVE
C       DEFINED.  ROTATING IT IN RELATIVE MODE BY TEN DEGREES.
C
        CALL UMOVE (0.0,0.0)
        CALL USET ('RELATIVE')
        CALL UPSET ('ROTATE',DEGREE)
        CALL UPLYGN (0.0,0.0,3.0,1.0)
        CALL USET ('ABSOLUTE')
        CALL UPRINT (-1.0,-1.0,DEGREE)
    1   CONTINUE
        CALL UEND
        STOP
        END
```

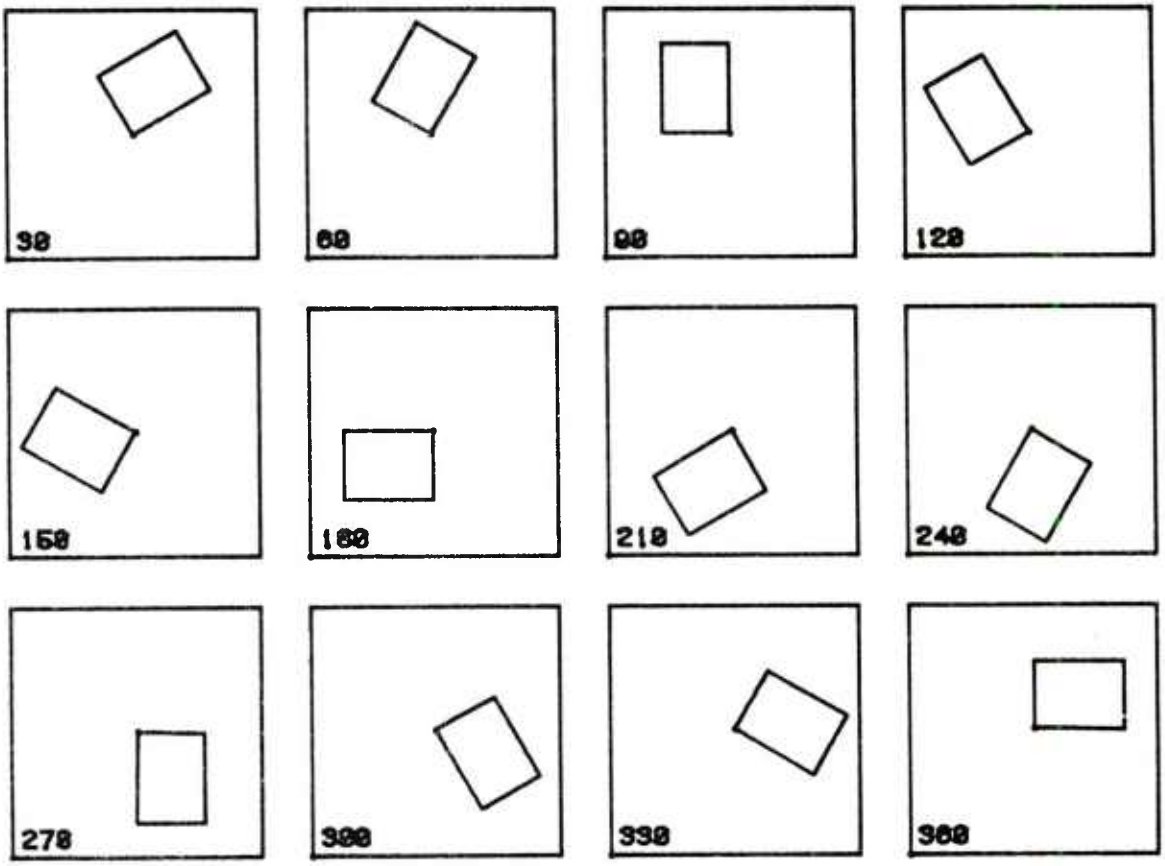
EXAMPLE VI-5




```
      DATA DEGREE,Y0/0.0,5.563/
      CALL USTART
      CALL UWINDO (-1.1,1.1,-1.1,1.1)
      CALL USET ('INTEGER')
      CALL USET ('LARROW')
      DO 1 I = 1, 3
      X0 = -1.5
      Y0 = Y0 - 1.8
      DO 1 J = 1, 4
      X0 = X0 + 1.8
      DEGREE = DEGREE + 10.0
      CALL UDAREA (X0,(X0+1.5),Y0,(Y0+1.5))
      CALL UOUTLN
      CALL UMOVE (0.0,0.0)
      CALL USET ('RELATIVE')
      CALL UPSET ('ROTATE',DEGREE)
      CALL UPLYGN (3.0,0.0,3.0,1.0)
      CALL USET ('ABSOLUTE')
      CALL UPRINT (-1.0,-1.0,DEGREE)
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO DEMONSTRATE USE OF SUBROUTINE 'URECT'.
C       A RECTANGLE WILL BE DRAWN IN RELATIVE MODE WITH A ROTATIONAL
C       INCREMENT OF THIRTY DEGREES APPLIED TO SUBSEQUENT CASES ON A
C       TEKTRONIX 4010/4013 TERMINAL.
C
        DATA DEGREE,YO/0.0,5.63/
C
C       ENTER GCS, SET VIRTUAL WINDOW, SET INTEGER MODE
C
        CALL USTART
        CALL UWINDO (-1.1,1.1,-1.1,1.1)
        CALL USET ('INTEGER')
C
C       LOOP THROUGH PROGRAM, CHANGING LOCATION OF FIGURE BY
C       INCREMENTING THE LOCATIO NBY A FIXED AMOUNT.
C
        DO 1 I=1,3
        XO=-1.5
        YO=YO-1.8
        DO 1 J=1,4
        XO=XO+1.8
        DEGREE=DEGREE+30.0
C
C       DEFINE DEVICE AREA SO THAT ONLY ONE RECTANGLE WILL BE DRAWN.
C
        CALL UDAREA (XO,(XO+1.5), YO,(YO+1.5))
        CALL UOUTLN
C
C       CALL 'URECT' TO DRAW THE RECTANGLE WITHIN THE DEVICE AREA WE
C       HAVE DEFINED, ROTATING IT IN RELATIVE MODE BY THIRTY DEGREES.
C
        CALL UMOVE (0.0,0.0)
        CALL USET ('RELATIVE')
        CALL UPSET ('ROTATE',DEGREE)
        CALL URECT (0.8,0.6)
        CALL USET ('ABSOLUTE')
        CALL UPRINT (-1.0,-1.0,DEGREE)
    1   CONTINUE
        CALL UEND
        STOP
        END
```

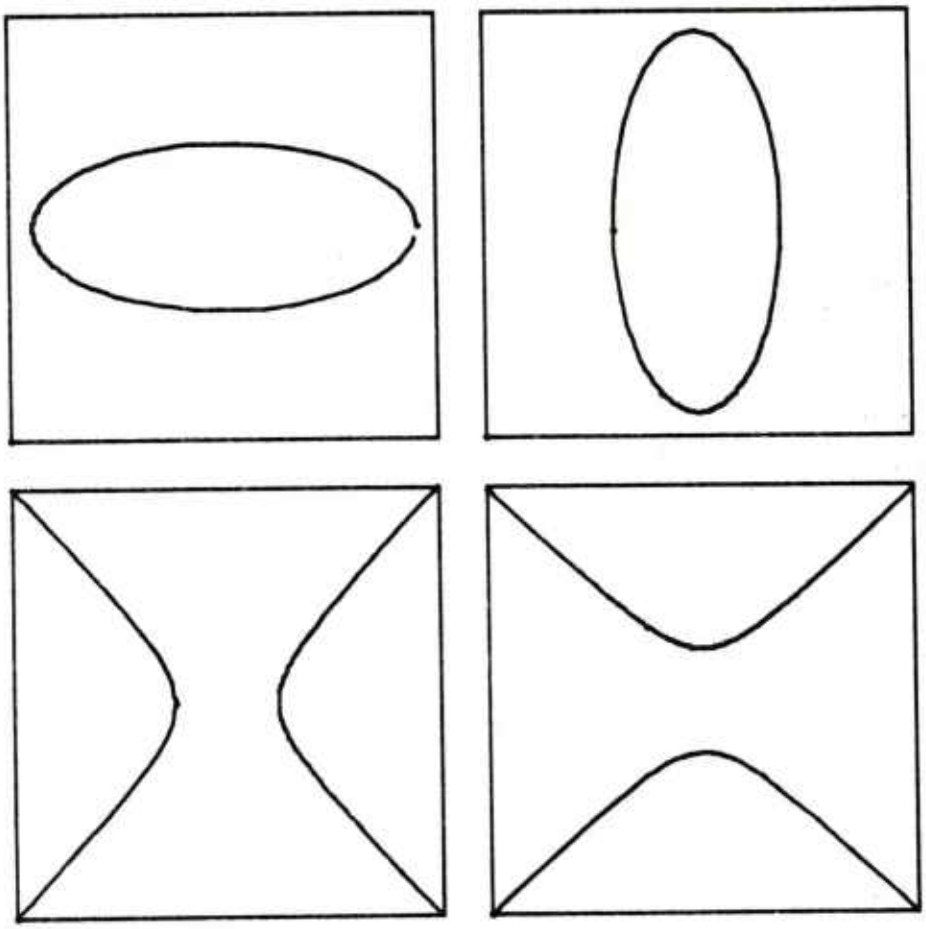
EXAMPLE VI-6




```
      DATA DEGREE,Y0/0.0,5.63/
      CALL USTART
      CALL UWINDO (-1.1,1.1,-1.1,1.1)
      CALL USET ('INTEGER')
      DO 1 I = 1, 3
      X0 = -1.5
      Y0 = Y0 - 1.6
      DO 1 J = 1, 4
      X0 = X0 + 1.6
      DEGREE = DEGREE + 30.0
      CALL UDAREA (X0,(X0+1.5),Y0,(Y0+1.5))
      CALL UOUTLN
      CALL UMOVE (0.0,0.0)
      CALL USET ('RELATIVE')
      CALL UPSET ('ROTATE',DEGREE)
      CALL URECT (0.6,0.6)
      CALL USET ('ABSOLUTE')
      CALL UPRINT (-1.0,-1.0,DEGREE)
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF 'UCONIC' TO
C       GENERATE ELLIPSES AND HYPERBOLAE. FOUR FIGURES WILL BE DRAWN:
C       2 ELLIPSES ORIENGED ALONG THE 'X' AND 'Y' AXES; AND 2 HYPERBOLAE
C       ORIENTED ALONG THE 'X' AND 'Y' AXES. EACH ONE OF THE FIGURES IS
C       DRAWN WITHIN ITS OWN REGION OF THE SCREEN BY REDEFINING THE
C       DEVICE AREA PRIOR TO DRAWING THE FIGURE ON A TEKTRONIX 4010/4013
C       TERMINAL.
C
C       SET UP DATA ARRAYS FOR UCONIC
C
        DIMENSION X(4),Y(4),P(4),E(4)
        DATA INDEX,YO,X,Y,P,E/
     *  0,5.73,10.,50.,67.,40.,50.,10.,50.,67,9.5-9.5,9.,-8.,.9,-.9,1.44,-1.44/
C
C       ENTER GCS
C
        CALL USTART
        DO 1 I=1,2
        XO=1.82
        YO=YO-2.86
        DO 1 J=1,2
        XO=XO+2.86
        INDEX=INDEX + 1
C
C       DEFINE DEVICE AREA SO THAT ONLY ONE FIGURE WILL BE DRAWN.
C
        CALL UDAREA (XO,(XO+2.57),YO,(YO+2.57)
        CALL UOUTLN
C
C       CALL UCONIC TO DRAW THE FIGURE USING THE PARAMETERS STORED IN
C       ARRAYS 'X','Y','P', AND 'E'. NOTE THAT DEFAULT VIRTUAL WINDOW IS
C       MAPPED TO THE CURRENT DEVICE AREA SPECIFICATION.
C
        CALL UCONIC (X(INDEX),Y(INDEX),P(INDEX),E(INDEX),0.0,360.0)
    1   CONTINUE
        CALL UEND
        STOP
        END
```

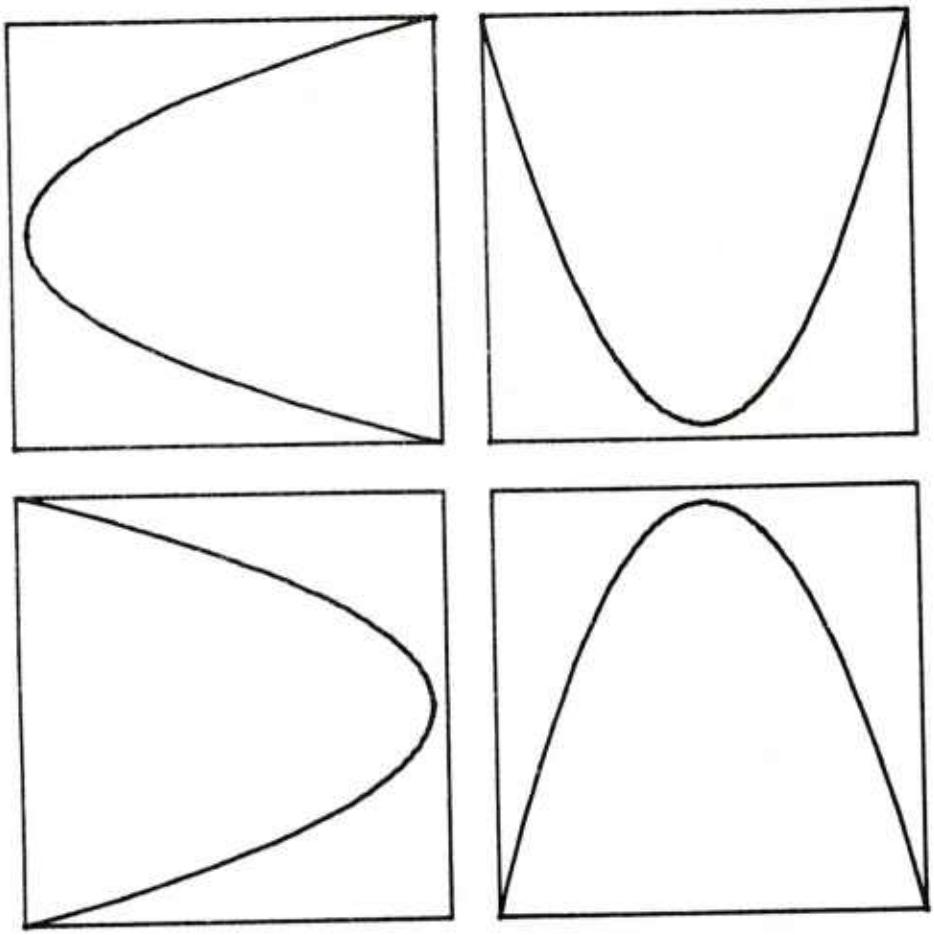
EXAMPLE VI-7



```
      DIMENSION X(4),Y(4),P(4),E(4)
      DATA INDEX,Y0,X,Y,P,E/0,5.73,10.,50.,67.,50.,50.,10.,50.,67.,
     &                      9.5,-9.5,9.,-9.,.9,-.9,1.44,-1.44/
      CALL USTART
      DO 1 I = 1, 2
      X0 = -1.62
      Y0 = Y0 - 2.66
      DO 1 J = 1, 2
      X0 = X0 + 2.66
      INDEX = INDEX + 1
      CALL UDAREA (X0,(X0+2.57),Y0,(Y0+2.57))
      CALL UOUTLN
      CALL UCONIC (X(INDEX),Y(INDEX),P(INDEX),E(INDEX),0.0,360.0)
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF 'UCONIC' TO
C       GENERATE PARABOLAE. FOUR FIGURES WILL BE DRAWN:  2 WILL BE
C       ORIENTED ALONG THE '+X' AND '+Y' AXES, AND 2 WILL BE ORIENTED
C       ALONG THE '-X' AND '-Y' AXES. EACH ONE OF THE PARABOLAE IS DRAWN
C       WITHIN ITS OWN REGION OF THE SCREEN BY REDEFINING THE DEVICE AREA
C       PRIOR TO DRAWING THE FIGURE ON A TEKTRONIX 4010/4013 TERMINAL.
C
        DIMENSION X(4),Y(4),P(4)
        DATA INDEX,YO,X,Y,P,E/
     &  0,5.73,10.,50.,90.,50.,50.,10.,50.,90.,13.,-13.,-13.,13.,1.,-1.,1.,-1./
C
C       ENTER GCS
C
        CALL USTART
        DO 1 I=1,2
        XO=1.82
        YO=YO-2.86
        DO 1 J=1,2
        XO=XO+2.86
        INDEX=INDEX+1
C
C       DEFINE DEVICE AREA SO THAT ONLY ONE FIGURE WILL BE DRAWN.
C
        CALL UDAREA (XO,(XO+2.57),YO,(YO+2.57))
        CALL UOUTLN
C
C       CALL UCONIC TO DRAW THE FIGURE USING THE PARAMETERS STORED IN
C       ARRAYS 'X','Y','P', AND 'E'.  NOTE THAT DEFAULT VIRTUAL WINDOW IS
C       MAPPED TO THE CURRENT DEVICE AREA SPECIFICATION.
C
        CALL UCONIC (X(INDEX),Y(INDEX),P(INDEX),E(INDEX),0.0,360.)
    1   CONTINUE
        CALL UEND
        STOP
        END
```

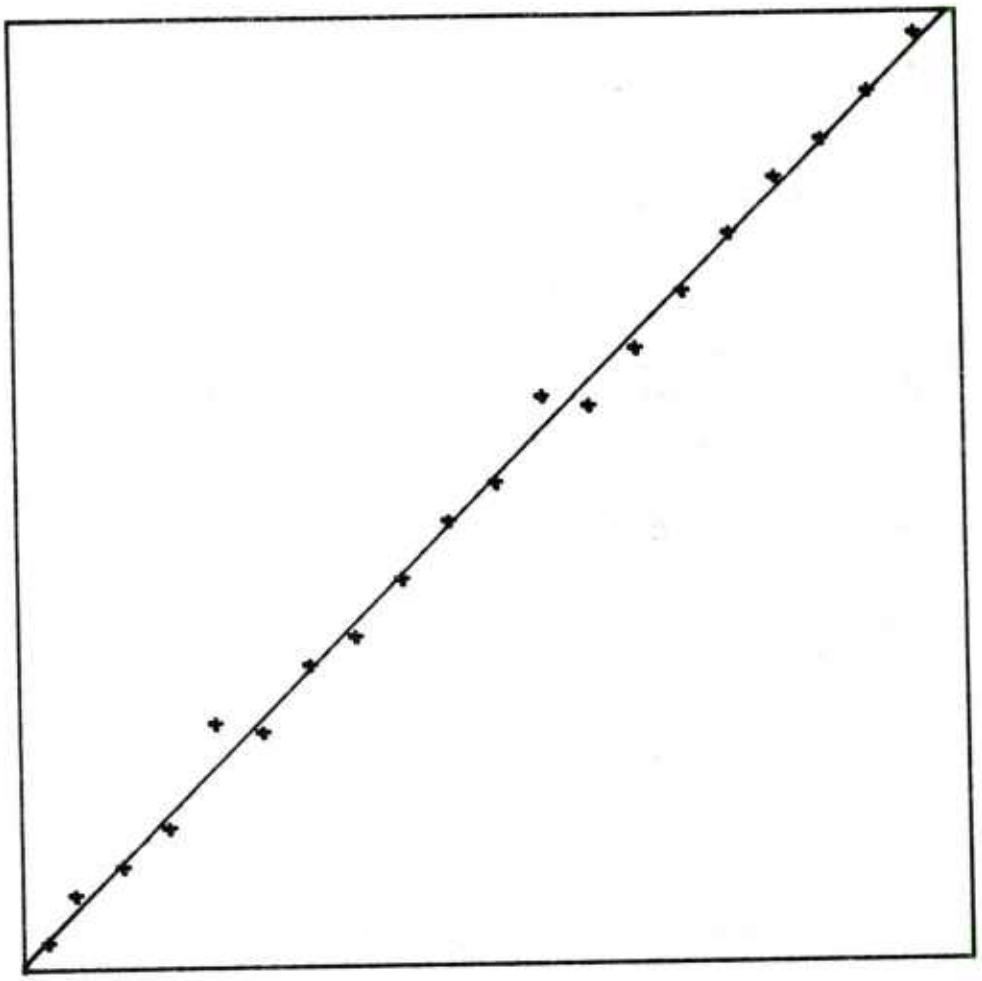
EXAMPLE VI-8



```
      DIMENSION X(4),Y(4),P(4),E(4)
      DATA INDEX,Y0,X,Y,P,E/0,5.73,10.,50.,90.,50.,50.,10.,50.,90.,
     &                      13.,-13.,-13.,13.,1.,-1.,1.,-1./
      CALL USTART
      DO 1 I = 1, 2
      X0 = -1.62
      Y0 = Y0 - 2.86
      DO 1 J = 1, 2
      X0 = X0 + 2.86
      INDEX = INDEX + 1
      CALL UDAREA (X0,(X0+2.57),Y0,(Y0+2.57))
      CALL UOUTLN
      CALL UCONIC (X(INDEX),Y(INDEX),P(INDEX),E(INDEX),0.0,360.0)
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF 'ULINFT' TO
C       CALCULATE THE SLOPE (S) AND Y-INTERCEPT (YI) OF A LINE WHICH
C       REPRESENTS THE 'BEST' LINEAR FIT TO A SERIES OF DATA POINTS.
C       DEFAULT VALUES OF VIRTUAL WINDOW, DEVICE AREA, AND LINE TYPE ARE
C       USED.
C
        DIMENSION X(20), Y(20)
        DATA X/0.,5., 10., 15., 20., 25., 30., 35., 40., 45., 50., 55., 60., 65., 70., 75., 80., 85.,
     &  90., 95./
        DATA Y/20., 25., 30., 35., 30., 25., 20., 15., 10., 15., 20., 25., 30., 35., 40., 45., 50., 55.,
     &  60., 65./
C
C       ENTER GCS, DRAW OUTLINE
C
        CALL USTART
        CALL UOUTLN
        CALL USET ('N+')
        DO 1 I=1,20
        XN=FLOAT(I)
C
C       AND PLOT EACH DATA POINT WITH A '+'.
C
        CALL UPEN (X(I),Y(I))
    1   CONTINUE
        CALL USET ('LINE')
C
C       CALL 'ULINFT' TO CALCULATE THE LINE'S SLOPE AND Y-INTERCEPT.
C
        CALL ULINFT (X,Y,XN,S,YI)
C
C       MOVE TO THE Y-INTERCEPT THEN GRAPH THE LINE USING YO=SX+YI.
C
        XMIN=0.0
        XMAX=100.0
        CALL UMOVE (XMIN,YI)
        YO=YI+S*MAX
        CALL UPEN (XMAX,YO)
        CALL UEND
        STOP
        END
```

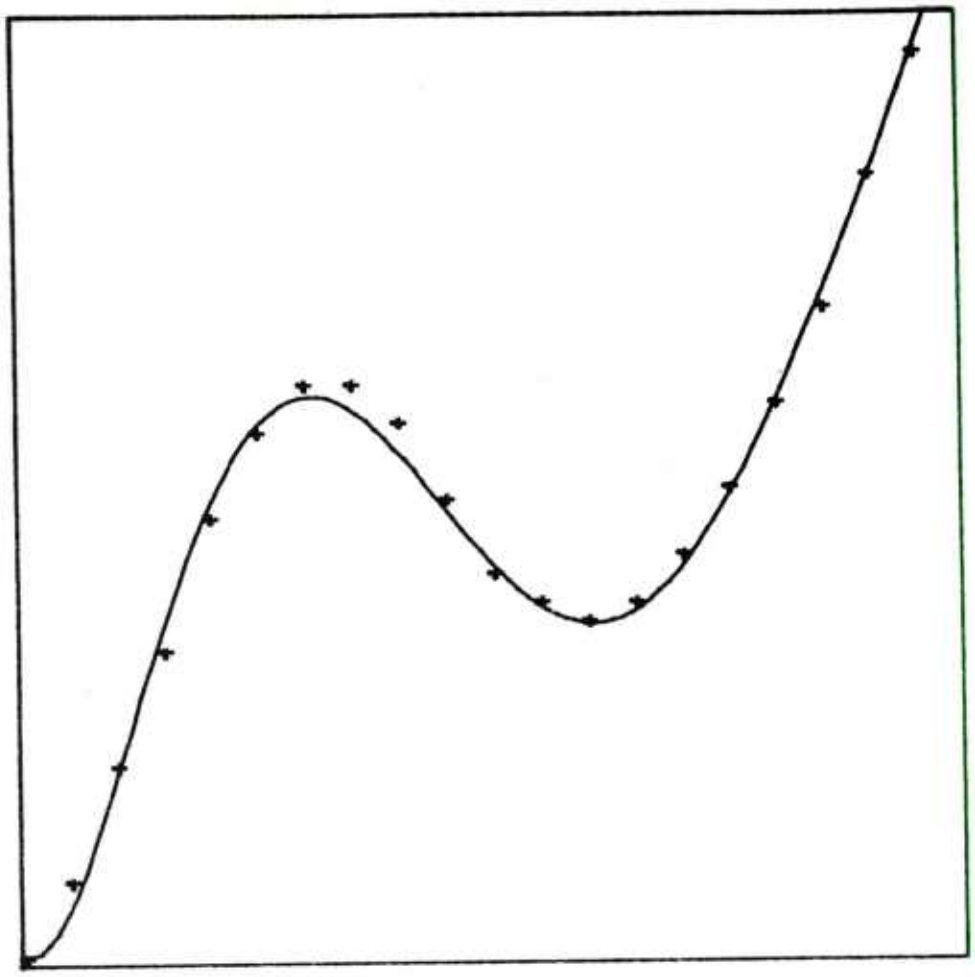
EXAMPLE VI-9



```
      DIMENSION X(20),Y(20)
      DATA X/2.,5.,10.,15.,20.,25.,30.,35.,40.,45.,
     &       50.,55.,60.,65.,70.,75.,80.,85.,90.,95./
      DATA Y/2.,7.,10.,14.,25.,24.,31.,34.,40.,46.,
     &       50.,59.,58.,64.,70.,76.,82.,86.,91.,97./
      CALL USTART
      CALL UOUTLN
      DO 2 I = 1, 20
      XN = FLOAT(I)
      CALL USET ('ACENTER')
      CALL USET ('N+')
      CALL UPEN (X(I),Y(I))
      CALL USET ('LINE')
    2 CONTINUE
      CALL ULINFT (X,Y,XN,S,YI)
      XMIN = 0.0
      XMAX = 100.0
      CALL UMOVE (XMIN,YI)
      Y0 = YI + S * XMAX
      CALL UPEN (XMAX,Y0)
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF 'ULSTSQ' TO
C       CALCULATE THE COEFFICIENTS OF A POLYNOMIAL OF ORDER IDEGRE-1
C       WHICH REPRESENTS THE 'BEST' FIT TO A SERIES OF DATA POINTS.
C       DEFAULT VALUES OF VIRTUAL WINDOW, DEVICE AREA, AND LINE TYPE ARE
C       USED. EACH OF THE DATA POINTS ARE PLOTTED WITH A '+'. AFTER
C       'ULSTSQ' IS CALLED, THE POLYNOMIAL IS THEN GRAPHED, USING THE
C       COEFFICIENTS WHICH WERE COMPUTED.
C
        PARAMETER IDEGRE=7
        DIMENSION A(IDEGRE),X(20),Y(20)
        DATA X/0.,5., 10., 15., 20., 25., 30., 35., 40., 45., 50.,
     *  55., 60., 65., 70., 75., 80., 85., 90., 95./
        DATA Y/20., 25., 30., 35., 30., 25., 20., 15., 10., 15., 20.,
     *  25., 30., 35., 40., 45., 50., 55., 60., 65./
C
C       ENTER GCS, DRAW OUTLINE, SET TYPE AND DEGREE OF FIT
C
        CALL USTART
        CALL UOUTLN
        CALL UPSET ('POLYNOMIAL',FLOAT(IDEGRE-1))
        CALL USET ('N+')
C
        PLOT POINTS
C
        DO 2 I=1,20
        CALL UPEN (X(I),Y(I))
    2   CONTINUE
C
C       MOVE PEN TO ORIGIN, COMPUTE LEAST SQUARES LINE
C
        CALL UMOVE (0.0, 0.0)
        CALL ULSTSQ (X,Y,20.,A)
C
C       PLOT LEAST SQUARES LINE APPROXIMATING POINTS
C
        DO 5 I=1,100
        YO=A(1)
        XO=FLOAT(I)
        XK=XO
        DO 4 J=2, IDEGREE
        YO=A(J)*XK+YO
        XK=XK*XO
    3   CONTINUE
        CALL UPEN (XO,YO)
    4   CONTINUE
        CALL UEND
        STOP
        END
```

EXAMPLE VI-10



```
      PARAMETER IDEGRE=7
      DIMENSION A(IDEGRE),X(20),Y(20)
      DATA X/0.,5.,10.,15.,20.,25.,30.,35.,40.,45.,
     &       50.,55.,60.,65.,70.,75.,80.,85.,90.,95./
      DATA Y/0.,8.,20.,32.,40.,55.,60.,60.,50.,48.,
     &       40.,37.,35.,37.,42.,49.,58.,66.,82.,95./
      CALL USTART
      CALL UOUTLN
      CALL UPSET ('POLYNOMIAL',FLOAT(IDEGRE-1))
      CALL USET ('ACENTER')
      CALL USET ('N+')
      DO 2 I = 1, 20
      CALL UPEN (X(I),Y(I))
    2 CONTINUE
      CALL USET ('LINE')
      CALL UMOVE (0.0,0.0)
      CALL ULSTSQ (X,Y,20.,A)
      DO 5 I = 1, 100
      Y0 = A(1)
      X0 = FLOAT(I)
      XK = X0
      DO 4 J = 2, IDEGRE
      Y0 = A(J) * XK + Y0
      XK = XK * X0
    4 CONTINUE
      CALL UPEN (X0,Y0)
    5 CONTINUE
      CALL UEND
      STOP
      END
```

CHAPTER VII

HIGH LEVEL GRAPHICS

High Level Plotting Facility

UPLOT is a general-purpose composite plotting subroutine which may be used to graph full-screen representations of complete arrays of data. Through the use of common default options, UPLOT provides sufficient flexibility to meet the majority of plotting requirements; however, should the user desire to change any of the applicable options, UPLOT will permit:

A. Rectangular, semi-logarithmic, log-log, or polar plotting;

B. Automatic or forced scaling of data values;

C. Axes presentation and format options, to include alphanumeric labeling.

UPLOT may be called by the following sequence:

CALL UPLOT (X,Y,CURVES, PARRAY,OPTARY)

CURVES is a single-valued variable which denotes the number of curves to be plotted. PARRAY is an array which describes the number of data points for each curve; e.g., PARRAY(1) is the number of data points comprising the first curve, PARRAY(2) is the number of points for the second curve, etc. OPTARY is a character array which specifies the data presentation format applicable to each given curve. Each element of OPTARY must not exceed 4 characters in length, and must represent a valid pen-status option such as 'LINE', 'VECT', etc. Since PARRAY and OPTARY follow the same general nomenclature in describing the characteristics of a given curve, both of these arrays must have as many elements as there are curves to be plotted. Arrays X and Y contain the actual data values grouped by curve. The size of the X and Y arrays must be equal to the sum of the elements of PARRAY. The general format of each of these arrays is outlined in the following figure.

Contents of Arrays Used in UPLOT

PURPOSE OF PLOT - Plot three curves having 8, 4 and 5 points each

	X	*Y*	*CURVES*	*PARRAY*	*OPTARY*
Curve 1	X_{11}	Y_{11}	3.	8.	*'LINE'*
	X_{12}	Y_{12}		4.	*'VECT'*
	X_{13}	Y_{13}		5.	*'DARR'*
	X_{14}	Y_{14}			
	X_{15}	Y_{15}			
	X_{16}	Y_{16}			
	X_{17}	Y_{17}			
	X_{18}	Y_{18}			
Curve 2	X_{21}	Y_{21}			
	X_{22}	Y_{22}			
	X_{23}	Y_{23}			
	X_{24}	Y_{24}			

Curve 3	X_{31}	Y_{31}
	X_{32}	Y_{32}
	X_{33}	Y_{33}
	X_{34}	Y_{34}
	X_{35}	Y_{35}

For those users desiring to plot only a single curve using the current line type, there exists a special form of UPLOT — UPLOT1 — which has a simplified calling sequence. UPLOT1 offers all of the various options available through UPLOT and may be called by the following sequence:

CALL UPLOT1 (X,Y,PTS)

X and Y are arrays containing the data values to be plotted and PTS is a single-valued variable indicating the number of points which comprise the curve.

The visual results obtained through UPLOT depend upon the status of the GSA at the time UPLOT is invoked. All of the high-level graphics parameters are assigned default values by USTART, and may be modified through conventional calls to USET and UPSET. Although many of these options will be presented, it is not the intent of this chapter to discuss each particular option in detail, further information may be found in the GCS Programmer's Reference Manual.

Coordinate System Options

The two types of coordinate systems applicable to UPLOT are specified through USET in the following manner:

```
CALL USET ('RECTANGULAR')
CALL USET ('POLAR')
```

Under RECTANGULAR coordinates, the user is given the option of specifying linear or logarithmic mappings of the independent and dependent variables: **NOLOGAXES**, XLOGAXES, YLOGAXES and XYLOGAXES. When logarithmic axes are specified, the user has the option of specifying the base of the logarithmic transform via a call to UPSET. This can be any real number, with 10. being default or the character string IHE to denote the base of the Napierian logarithm. The call to UPSET is

CALL UPSET ('BASE' *base)*

where base is the *base* of the logarithms as described above.

Scaling Options

There are three options applicable to the scaling of data values for a graphical display under UPLOT: **'AUTOSCALE'**, 'FULLSCALE', and 'OWNSCALE'. Under AUTOSCALE, the range of values to be plotted is examined and scaling occurs so that the graph will be presented with 'nice' numbers at the tic marks on the axis (if requested). The FULLSCALE option resembles AUTOSCALE with the exception that 'nice' numbers may not appear. OWNSCALE uses the virtual window established by the user upon entry to UPLOT in order to scale the data values, hence, a zero-value is not forced on the X-axis. Thus, 'nice' numbers may not necessarily appear as numeric labels, and all of the data points may not be displayed should they extend beyond the window limits.

There are also two options relating to the calculation of the scale factors, 'NEWSCALE' and 'OLDSCALE'. With the 'NEWSCALE' option, the scale factors used to map the data onto the display screen is always recalculated. Specifying 'OLDSCALE' results in this calculation being bypassed, with the result that a scaling factor used in a previous plot is used for the current plot. This ofter results in an increase in speed at which the p. ›ts are produced.

General Purpose Axes Creation

Subroutine **UAXIS** is invoked by subroutine **UPLOT** to generate the desired axes for the input data arrays. It is specified:

CALL UAXIS(XMIN,XMAX,YMIN,YMAX)

Axes Options

The options which apply to the axes may be further subdivided into the following categories:

A. Axes existence options.

B. Axes format options.

C. Axes positioning options

D. X and Y labeling option.

E. Numeric label format options.

'XYAXES', 'XAXIS', 'YAXIS', and 'NOAXES' comprise the USET options which specify the existence of an axis XYAXES indicates that both X and Y axes are to be drawn, whereas the XAXIS or YAXIS options are be used when only one of the respective axes is desired. The NOAXES option will suppress the display of the X and Y axes, but will permit axis labelling should the user so desire. The existence or lack of any axis does not influence the scaling mode which the user has requested.

The axes format options consist of allowing the user to specify either 'PLAINAXES', **'TICAXES'**, or 'GRIDAXES'. Under PLAINAXES, the specified axis (or axes) will appear as a solid line. The TICAXES option will force the axes to be drawn as ticked lines, with the tic intervals specified by the UPSET options:

CALL UPSET ('TICX',XINTERVAL)
CALL UPSET ('TICY',YINTERVAL)

XINTERVAL and YINTERVAL are single-valued REAL variables which specify the interval (in current units) at which tic marks are to appear on the X and Y axes respectively. Under the default values of zero X and Y tick intervals, UPLOT will choose suitable intervals based upon the axes existence and data scaling options specified by the user. The GRIDAXES option should be used whenever grid lines are desired for the axes which have been defined. The grid spacing will be determined in the same manner as the tic interval under TICAXES. Under POLAR coordinates, XINTERVAL defines the radial distance between concentric circles which comprise the grid lines for the R-axis; whereas, YINTERVAL specifies the angular increment at which grid lines which be drawn for the 0-axis.

The user may position his axes through the use of the following options: **'EDGEAXES'**, 'ZEROAXES', 'XZEROYEDGE', 'YEDGEXZERO', 'XEDGEYZERO', and 'YZEROXEDGE'. The EDGEAXES option indicates that the axis (or axes) is to be oriented near the edges of the currently defined device area. When the EDGEAXIS option is used in conjunction with XYAXES and OWNSCALE, UPLOT will generate X and Y axes which intersect at the minimum values contained within the X and Y arrays. In addition, the axes will be positioned near the left and bottom edges of the device area. Should the minimum and maximum values of the X and Y arrays constitute an interval which contains zero, the ZEROAXES option may be used to force the intersection of the axes through the given zero point. Should ZEROAXES be specified, and zero is not contained within the defined interval, EDGEAXES will be generated. XZEROYEDGE, YEDGEXZERO, XEDGEYZERO, and YZEROXEDGE options are merely extensions of the EDGEAXES and ZEROAXES options, and are subject to the constraints outlined above.

The user is provided with the ability to control the format under which the X and Y axes are labeled through the use of the following options: 'NOXLABEL', **'XNUMERICLABEL'**, 'XALPHALABEL', 'XBOTHLABELS', 'NOYLABEL', **'YNUMERICLABEL'**, 'YALPHALABEL', and 'YBOTHLABELS'. 'NOXLABEL' and 'NOYLABEL' should be used when the labeling of the X and Y axes is to be supressed. The 'XNUMERICLABEL' and 'YNUMERICLABEL' options indicate that numeric labels are to be created along the X and Y axes at the tic intervals specified by 'TICX' and 'TICY' respectively. The actual values for the numeric labels will be based upon the axes scaling options previously discussed. 'XALPHANUMERIC' and 'YALPHANUMERIC' allow the user to generate alphanumeric labels (or titles) for each axis. The actual information to be displayed is defined through a call to UPSET:

CALL UPSET ('XLABEL',XDATA)
CALL UPSET ('YLABEL',YDATA)

XDATA and YDATA are strings of up to 40 characters in length, with the standard string termination character inserted at the end of the information. The XBOTHLABELS and YBOTHLABELS options are used when numeric as well as alphabetic labels are desired.

The user is provided with the ability to control the format under which numeric labels will be printed through the use of the **'BESTFORMAT'**, 'IFORMAT', and 'GFORMAT' options. Under 'IFORMAT', 'EFORMAT', all numeric labels will appear as integers, with truncation occurring for any non-integral values. EFORMAT and GFORMAT will generate labels which will be edited into either standard FORTRAN 'E' or 'G' formats respectively. BESTFORMAT implies that the numeric labels are to be output under the format which is most appropriate for the given label.

Time Series Axes

In addition, there is the axes routine, UTAXIS, that permits the user to select, scale and label time series axes. For further information on this routine, see the subroutine description section. It is specified:

CALL UTAXIS(BEGPER,PERIOD,YMIN,YMAX)

Curve Fitting Options

UPLOT provides an additional facility for those users desiring to fit linear, least-squares polynominal, and spline curves to their data points. Interested users are directed to the detailed descriptions of UPLOT, ULSTSQ, and ULINFT contained within GCS programmer's reference manual.

Extended High-Level Graphics

For data presentation formats extending beyond those available through UPLOT and UPLOT1, a series of generalized histogram, barchart and piechart utility subroutines have been incorporated into GCS. These subroutines are comprehensive enough to permit their utilization as the core of a complete data analysis and display program; without compromising the versatility to be incorporated into a graphics program within which they play a relatively minor role.

Histogram generation facilities are provided through the use of subroutine UHISTO:

CALL UHISTO (DATA,XN,CELLS)

DATA is a REAL array containing the data values from which the histogram is to be constructed, XN is a single-valued REAL variable which is used to specify the number of elements in the DATA array; and CELLS is a single-valued REAL variable which designates the number of cells (or bars) which are to appear on the histogram. The standard axes existance, labeling and scaling options previously discussed for UPLOT also apply to UHISTO, with the exception of the GRIDAXIS option for the X and Y axes, and 'TICAXIS' for the Y axis.

Subroutine UBAR may be invoked whenever the user desires to generate a barchart.

CALL UBAR (DATA,XN,LABELS,SLABEL)

DATA and XN are interpreted in the same manner as described above the UHISTO. LABELS is a character array which is used to specify the label for each XN values stored in DATA. SLABEL is a single-valued REAL variable which defines the length (in characters) of each element of the LABELS array. Should UBAR encounter the standard string termination character (;) while processing any of the label items, no additional characters are obtained from the item and the string prior to the terminator is scaled to a string of SLABEL characters in length.

Subroutine UCHART may be invoked whenever the user desires to gnerate a grouped bar chart for multi-valued data.

CALL UCHART(ARRAY,GROUPS,BARS,LABELS,YMAXL)

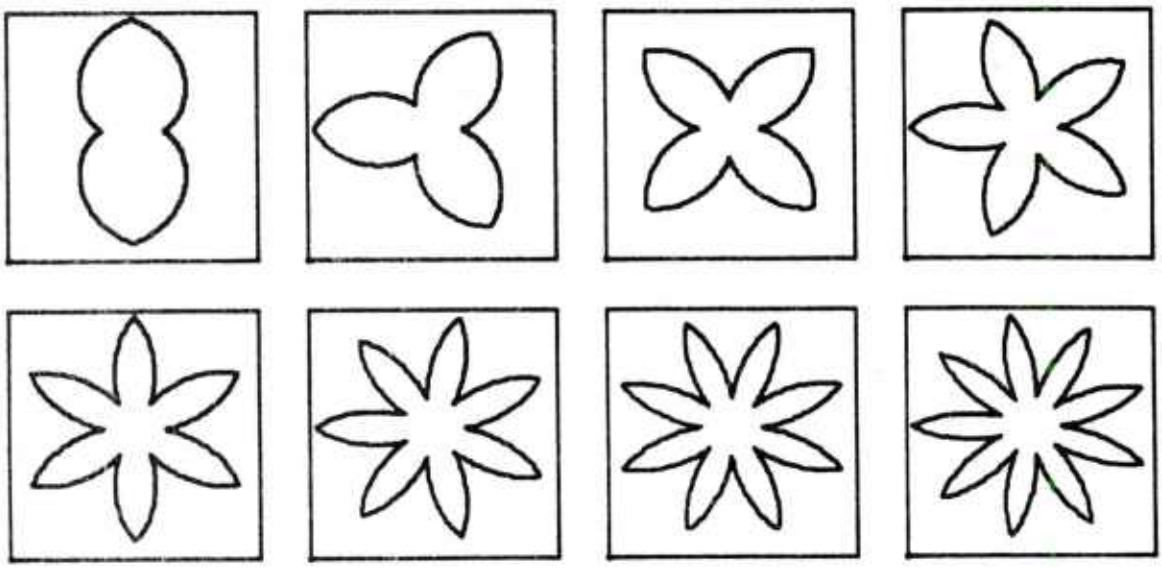
Standard piechart generation facilities are available under GCS through subroutine UPIE which may be called by the following sequence:

CALL UPIE (DATA,XN,LABELS,SLABEL)

DATA, XN LABELS and SLABEL are interpreted in the same manner as under UBAR.

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF SUBROUTINE
C       'UPLOT1'.  EIGHT CURVES WILL BE PLOTTED IN POLAR COORDINATES,
C       USING THE SINGLE-CURVE POLAR PLOTTING ROUTINE.  THE PURPOSE OF
C       THIS EXAMPLE IS TO ILLUSTRATE THE USE OF THE HIGH-LEVEL
C       SUBROUTINE TO PLOT SINGLE CURVES WITHIN THE USER-DEFINED DEVICE
C       AREA.  NO AXES OR AXES LABELING IS REQUESTED.  THIS WAS WRITTEN
C       FOR A TEKTRONIX 4010/4013 TERMINAL.
C
C       INITIALIZE CONSTANTS AND ANGLES
C
        DIMENSION R(361), THETA(361)
        DATA PI,W,YO/3.1415925,0.0,4.71/
        DO 1 I=1,361
    1   THETA(I)=FLOAT(I) *PI/180.0
C
C       ENTER GCS, SET FOR POLAR, RADIAN MEASURE, NO AXES AND NO LABELS.
C
        CALL USTART
        CALL USET ('POLAR')
        CALL USET ('RADIANS')
        CALL USET ('NOAXES')
        CALL USET ('NOXLABEL')
C
C       INCREMENT LOCATION OF DEVICE PLOTTING AREA, AND PLOT THE.
C       FIGURES.
C
        DO 3 I=1,2
        XO=-1.5
        YO=YO-1.8
        DO 3 J=1,4
        W=W+0.5
        XO=XO+1.8
        CALL UDAREA (XO,(XO+1.5),YO,(YO+1.5))
        CALL UOUTLN
        DO 2 K=1,361
    2   R(K)=1.2-0-.7*(ABS(COS(W*THETA(K)))-ABS(SIN(W*THETA(K))))
C
C       PLOT 361 DATA POINTS SPECIFIED BY 'R' AND 'THETA' USING THE
C       STANDARD LINE OPTION WITHIN THE DEVICE AREA WE HAVE DEFINED.
C       NOTE THAT NO AXES OR AXES LABELING WAS SPECIFIED.
C
        CALL UPLOT1 (R,THETA,361.0)
    3   CONTINUE
        CALL UEND
        STOP
        END
```

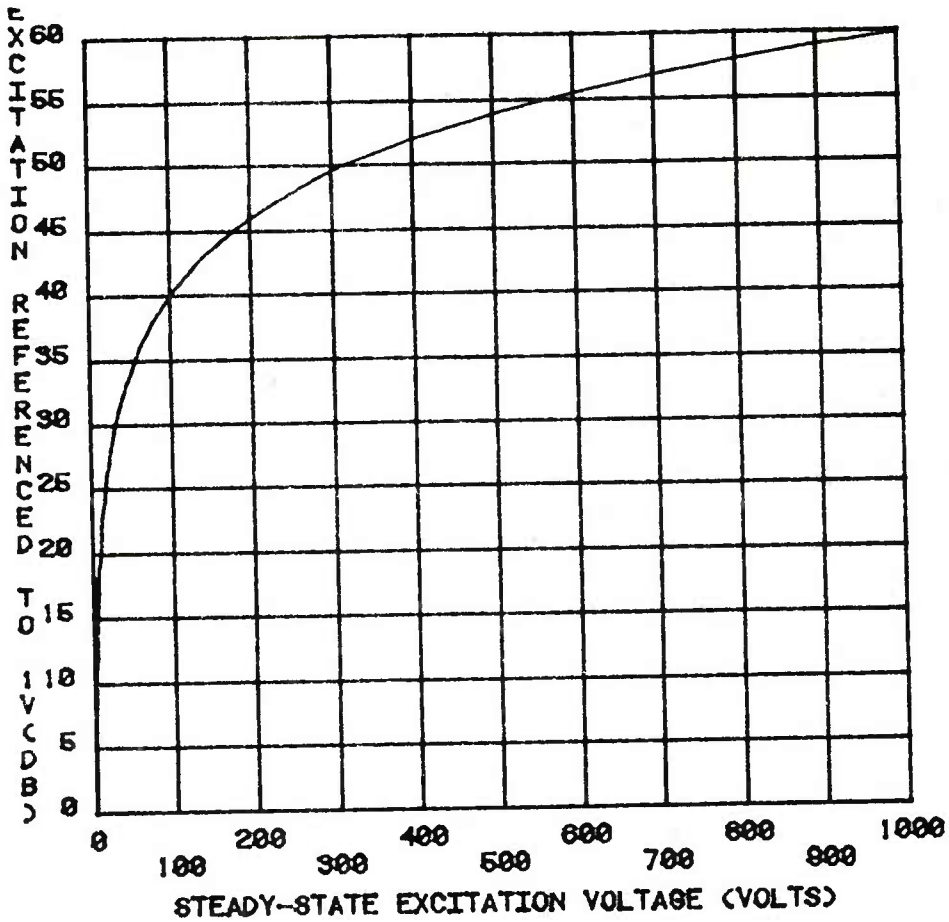
EXAMPLE VII-1



```
      DIMENSION R(361),THETA(361)
      DATA PI,W,Y0/3.14159265,0.5,4.71/
      DO 1 I = 1, 361
    1 THETA(I) = FLOAT(I) * PI / 180.0
      CALL USTART
      CALL USET ('POLAR')
      CALL USET ('RADIANS')
      CALL USET ('NOAXES')
      CALL USET ('NOXLABEL')
      CALL USET ('NOYLABEL')
      DO 3 I = 1, 2
      X0 = -1.5
      Y0 = Y0 - 1.8
      DO 3 J = 1, 4
      W = W + 0.5
      X0 = X0 + 1.8
      CALL UDAREA (X0,(X0+1.5),Y0,(Y0+1.5))
      CALL UOUTLN
      DO 2 K = 1, 361
    2 R(K) = 1.2-0.7*(ABS(COS(W*THETA(K)))-ABS(SIN(W*THETA(K))))
      CALL UPLOT1 (R,THETA,361.0)
    3 CONTINUE
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF SUBROUTINE
C       'UPLOT1'. DATA VALUES CONTAINED IN ARRAYS 'E2' AND 'DB' ARE PLOTTED
C       IN A CARTESIAN COORDINATE SYSTEM. A GRIDDED AXES, AND
C       ALPHANUMERIC AXES LABELING ARE PROVIDED AS ADDITIONAL OPTIONS.
C       DEFAULT VALUE OF DEVICE AREA IS USED.
C
C       LOAD LABELS INTO ARRAYS, INITIALIZE ARRAYS.
C
        DIMENSION DB(61),E2(61), ARRAY(8)
        CHARACTER XLABEL*40,YLABEL*40
        DATA XLABEL/'STEADY-STATE EXCITATION VOLTAGE (VOLTS);'/
        DATA YLABEL/'EXCITATION REFERENCED TO 1V (DB);'/
        DO 1 I=1,61
        DB(I)=FLOAT(I-1)
    1   E2(I)=10.0**(DB(I)/20.0)
C
C       ENTER GCS
C
        CALL USTART
        CALL USTUD (ARRAY)
C
C       INDICATE THE DATA TO BE USED FOR AXES LABELING INFORMATION.
C
        CALL UPSET ('XLABEL',XLABEL)
        CALL UPSET ('YLABEL',YLABEL)
        CALL UPSET ('TERMINATOR',';')
C
C       INDICATE THAT GRIDDED AXES, TOGETHER WITH NUMERIC AND ALPHA
C       LABELS ARE DESIRED.
C
        CALL USET ('GRIDAXES')
        CALL USET ('XBOTHLABEL')
        CALL USET ('YBOTHLABEL')
C
C       CALL 'UPLOT1' TO PLOT A SINGLE CURVE OF 61 POINTS USING 'E2' AND 'DB'.
C       STANDARD LINE OPTION IS REQUESTED.
C
        CALL UPLOT1 (E2,DB,61.0)
C
C       USE CROSSHAIRS TO DEFINE LOWER BOUNDARY THEN UPPER BOUNDARY.
C       THEN CALL UWINDO AND PLOT JUST THAT PORTION.
C
        CALL UGRIN (XL,YL,IC)
        CALL UGRIN (XU,YU,IC)
        CALL UWINDO (XL,XI,YL,YU)
        CALL USET ('OWNSCALE')
        CALL UDAREA (ARRAY(5), ARRAY(6), ARRAY(7), ARRAY(8)
        CALL UERASE
        CALL UPLOT1 (E2,DB,61.)
        CALL UEND
        STOP
        END
```

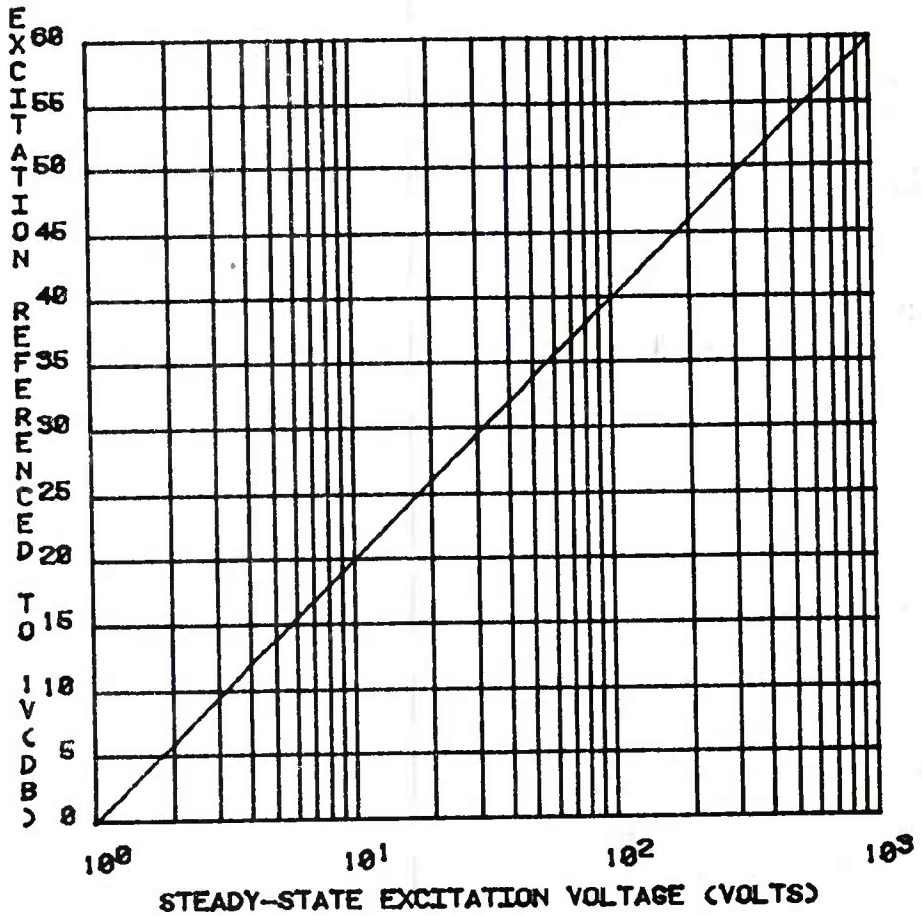
EXAMPLE VII-2




```
      DIMENSION DB(61),E2(61)
      CHARACTER XLABEL*40,YLABEL*40
      DATA XLABEL/'STEADY-STATE EXCITATION VOLTAGE (VOLTS);'/
      DATA YLABEL/'EXCITATION REFERENCED TO 1V(DB);'/
      DO 1 I = 1, 61
      DB(I) = FLOAT(I-1)
    1 E2(I) = 10.0**(DB(I) / 20.0)
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL UPSET ('XLABEL',XLABEL)
      CALL UPSET ('YLABEL',YLABEL)
      CALL USET ('GRIDAXES')
      CALL USET ('XBOTHLABEL')
      CALL USET ('YBOTHLABEL')
      CALL UPLOT1 (E2,DB,61.0)
      CALL UEND
      STOP
      END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF SUBROUTINE
C     'UPLOT1'.  DATA VALUES CONTAINED IN ARRAYS 'E2' AND 'DB' ARE PLOTTED
C     IN A SEMI-LOGARITHMIC COORDINATE SYSTEM.  A GRIDDED AXES, AND
C     ALPHANUMERIC AXES LABELING ARE PROVIDED AS ADDITIONAL OPTIONS.
C     DEFAULT VALUE OF DEVICE AREA IS USED.
C
C     INITIALIZE ARRAYS, LOAD LABELS INTO LABEL ARRAYS.
C
      DIMENSION DB(61),E2(61)
      CHARACTER XLABEL*40,YLABEL*40
      DATA XLABEL/'STEADY-STATE EXCITATION VOLTAGE (VOLTS);'/
      DATA YLABEL/'EXCITATION REFERENCED TO 1V (DB);'/
      DO 1 I=1,61
      DB(I)=FLOAT(I-1)
    1 E2(I)=10.0**(DB(I)/20.0)
C
C     ENTER GCS
C
      CALL USTART
C
C     INDICATE THE DATA TO BE USED FOR AXES LABELING INFORMATION.
C
      CALL UPSET ('TERMINATOR', ';')
      CALL UPSET ('XLABEL',XLABEL)
      CALL UPSET ('YLABEL',YLABEL)
C
C     INDICATE THAT GRIDDED AXES, TOGETHER WITH NUMBERIC AND ALPHA
C     LABELS ARE DESIRED.  ALSO SPECIFY WHICH AXIS IS LINEAR, AND WHICH IS
C     LOGARITHMIC.
C
      CALL USET ('GRIDAXES')
      CALL USET ('XBOTHLABEL')
      CALL USET ('YBOTHLABEL')
      CALL USET ('LOGXAXIS')
C
C     CALL 'UPLOT1' TO PLOT A SINGLE CURVE OF 61 POINTS USING 'E2' AND 'DB'.
C     STANDARD LINE OPTION IS REQUESTED.
C
      CALL UPLOT1 (E2,DB,61.0)
      CALL UEND
      STOP
      END
```

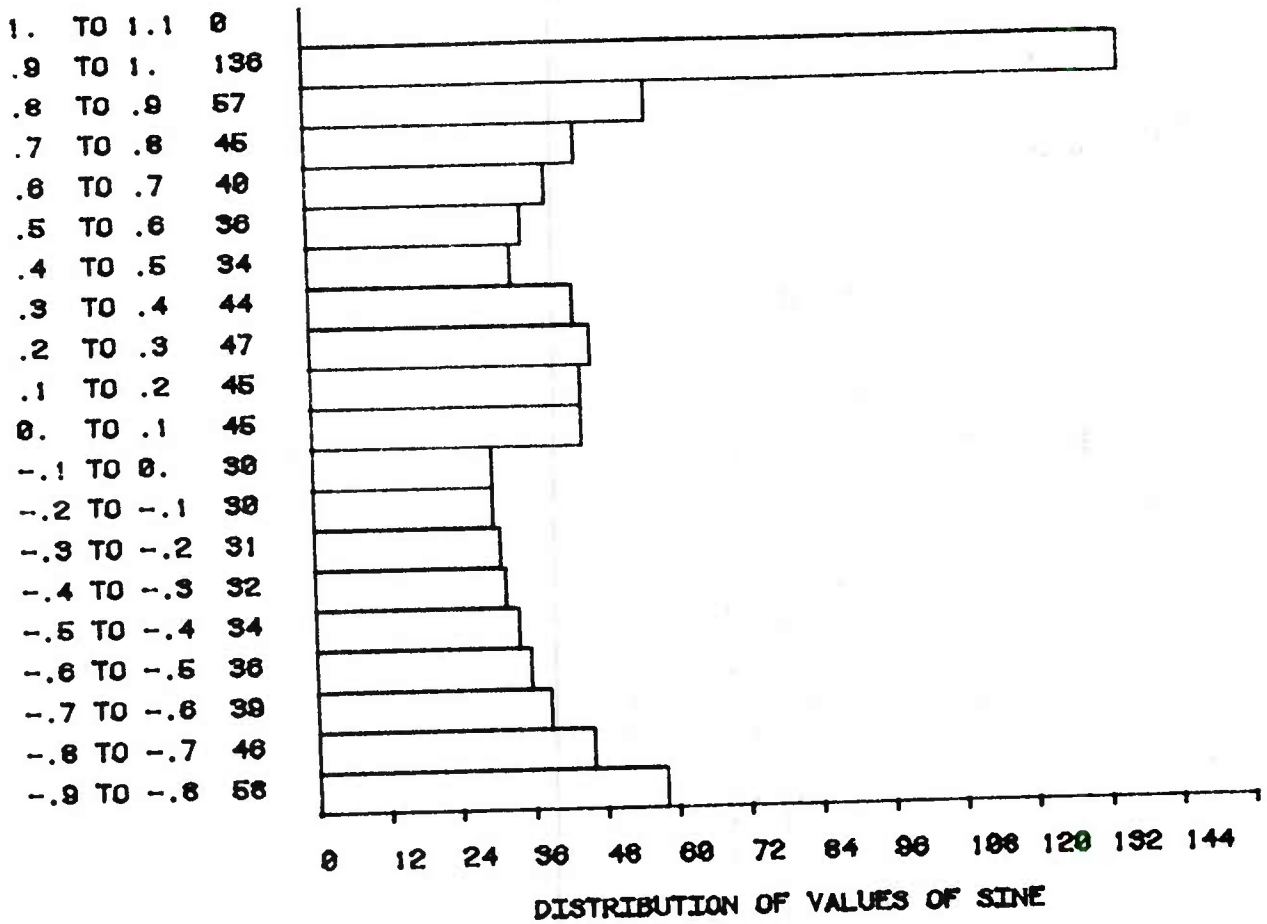
EXAMPLE VII-3




```
      DIMENSION DB(61),E2(61)
      CHARACTER XLABEL*40,YLABEL*40
      DATA XLABEL/'STEADY-STATE EXCITATION VOLTAGE (VOLTS);'/
      DATA YLABEL/'EXCITATION REFERENCED TO 1V(DB);'/
      DO 1 I = 1, 61
      DB(I) = FLOAT(I-1)
    1 E2(I) = 10.0**(DB(I) / 20.0)
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL UPSET ('XLABEL',XLABEL)
      CALL UPSET ('YLABEL',YLABEL)
      CALL USET ('GRIDAXES')
      CALL USET ('XBOTHLABEL')
      CALL USET ('YBOTHLABEL')
      CALL USET ('XLOGAXIS')
      CALL USET ('LINYAXIS')
      CALL UPLOT1 (E2,DB,61.0)
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM TO ILLUSTRATE APPLICATION OF GCS SUBROUTINE
C       'UHISTO'.
C
C       ALLOCATE AN ARRAY TO STORE DATA VALUES, AND INITIALIZE A COUNTER.
C
        DIMENSION DATA (1000)
        DATA XN/1000./
C
C       BUILD DATA FROM SINE VALUES
C
        DO 2 I=1,1000
    2   DATA(I) = SIN(FLOAT(I)/150.)
C
C       ENTER GCS AND INDICATE THAT THE ENTIRE DEVICE AREA IS DESIRED.
C
        CALL USTART
        CALL UPSET ('TERMINATOR', ';')
        CALL USET ('LARGE')
        CALL USET ('PERCENTUNITS')
        CALL UDAREA (0.,100.,0.,100.)
C
C       INDICATE THAT THE HISTOGRAM IS TO BE 'FULLSCALED' AND THAT THE X
C       AXIS IS TO HAVE ALPHABETIC AS WELL AS NUMERIC LABELS. ALSO SET
C       THE ALPHABETIC LABEL WHICH IS TO BE PRINTED ON THE X AXIS.
C
        CALL USET ('FULLSCALE')
        CALL USET ('XBOTHLABELS')
        CALL USET ('XLABEL', 'DISTRIBUTION OF VALUES OF SINE;')
C
C       PROCESS THE DATA ARRAY USING 20 CELLS.
C
        CALL UHISTO (DATA,XN,20.)
C
C       TERMINATION
C
        CALL UEND
        STOP
        END
```

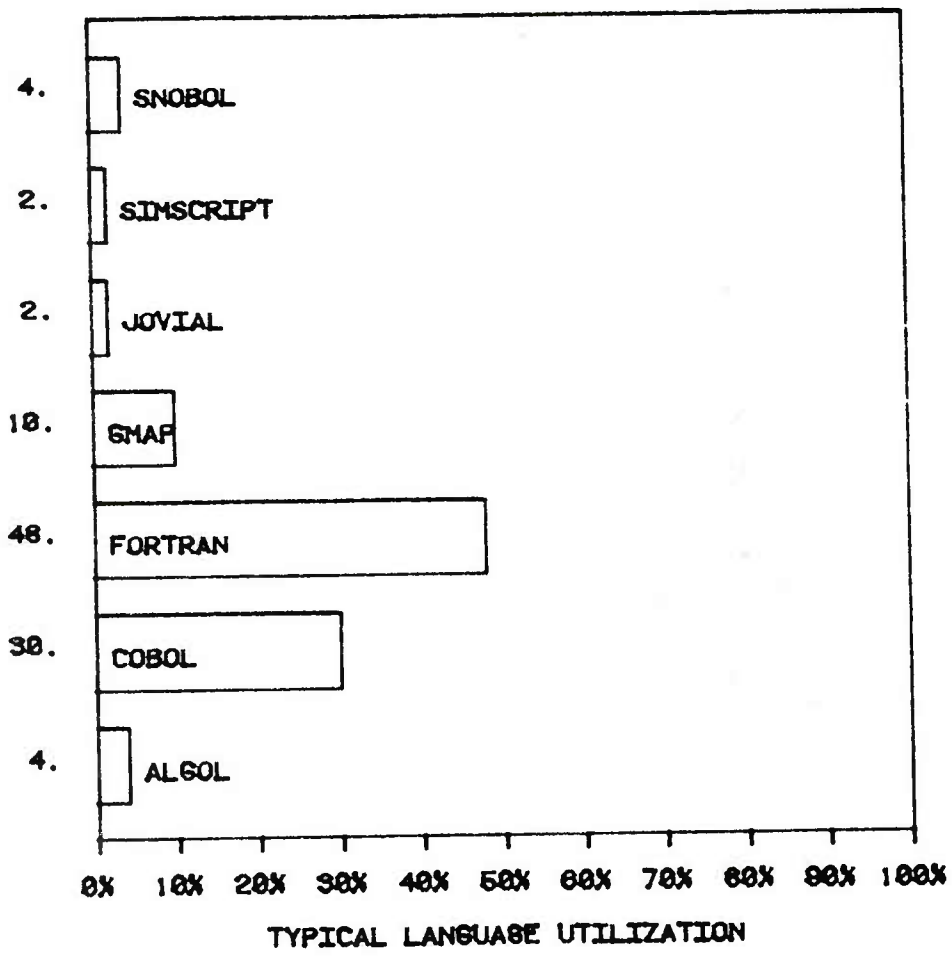
EXAMPLE VII-4




```
      DIMENSION DATA(1000)
      DATA XN/1000./
      DO 2 I = 1, 1000
    2 DATA(I) = SIN(FLOAT(I)/150.)
      CALL USTART
      CALL USET ('LARGE')
      CALL USET ('PERCENTUNITS')
      CALL UDAREA (0.,100.,0.,100.)
      CALL USET ('FULLSCALE')
      CALL USET ('XBOTHLABELS')
      CALL UPSET ('TERMINATOR',',')
      CALL UPSET ('XLABEL','DISTRIBUTION OF VALUES OF SINE;')
      CALL UHISTO (DATA,XN,20.)
      CALL UEND
      STOP
      END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF GCS
C     SUBROUTINE 'UBAR' TO DISPLAY A BAR-CHART.
C
C     ALLOCATE ARRAYS TO STORE DATA VALUES AND LABELS FOR THE CHART.
C
      DIMENSION DATA (7)
      CHARACTER LABELS*12 (7)
      DATA DATA,Y/4.,30.,48.,10.,2.,2.,4.,100./
      DATA LABELS/'ALGOL;' 'COBOL;', 'FORTRAN;' 'GMAP;', 'JOVIAL;',
     *     'SIMSCRIPT;', 'SNOBOL;'/
C
      CALL USTART
      CALL UPSET ('TERMINATOR', ';')
      CALL USET ('XBOTHLABELS')
      CALL UPSET ('XLABEL', 'TYPICAL LANGUAGE UTILIZATION AT USMA;')
C
C     CALL UBAR TO DISPLAY THE DATA VALUES CONTAINED WITHIN THE DATA
C     ARRAY, AND LABELS SPECIFIED BY THE LABELS ARRAY.
C
      CALL UBAR (DATA,FLOAT(NUMBER),LABELS,12.)
C
C     WRAP-UP GRAPHICS ACTIVITY AND TERMINATE THIS FORTRAN PROGRAM.
C
      CALL UEND
      STOP
      END
```

EXAMPLE VII-5




```
PARAMETER NUMBER=7
DIMENSION DATA(NUMBER)
CHARACTER LABELS*12(NUMBER)
DATA DATA/4.,30.,48.,10.,2.,2.,4./
DATA LABELS/'ALGOL;','COBOL;','FORTRAN;','GMAP;','JOVIAL;',
&           'SIMSCRIPT;','SNOBOL;'/
CALL USTART
CALL UPSET ('TERMINATOR',';')
CALL USET ('XBOTHLABELS')
CALL UPSET ('XLABEL','TYPICAL LANGUAGE UTILIZATION;')
CALL UBAR (DATA,FLOAT(NUMBER),LABELS,12.)
CALL UEND
STOP
END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF GCS
C       SUBROUTINE 'UPIE'.  IN ADDITION TO USING UPIE TO DISPLAY A PIE-CHART,
C       SOME ADDITIONAL GRAPHIC OUTPUT IS PERFORMED THROUGH USE OF
C       UPRNT1.
C
C       ALLOCATE ARRAYS TO STORE DATA VALUES AND LABELS FOR THE CHART.
C
        DIMENSION DATA (7)
        CHARACTER LABELS*12 (7)
        DATA DATA,Y/4.,30.,48.,10.,2.,2.,4.,100./
        DATA LABELS/'ALGOL;' 'COBOL;', 'FORTRAN;' 'GMAP;', 'JOVIAL;',
     *      'SIMSCRIPT;', 'SNOBOL;'/
C
C       ENTER GCS, INDICATE THAT AN ALPHABETIC LABEL IS DESIRED FOR THE X
C       AXIS, AND SPECIFY THE LABEL WHICH IS TO BE PRINTED.
C
        CALL USTART
        CALL UPSET ('TERMINATOR', ';')
        CALL USET ('XALPHABETIC')
        CALL UPSET ('XLABEL', 'TYPICAL LANGUAGE UTILIZATION AT USMA;')
C
C       CALL UPIE TO DISPLAY THE DATA VALUES CONTAINED WITHIN THE DATA
C       ARRAY, AND LABELS SPECIFIED BY THE LABELS ARRAY.
C
        CALL UPIE (DATA,FLOAT(NUMBER),LABELS,12.)
C
C       SET ADDRESSING MODE TO 'DEVICE'/'PERCENTUNITS'. SET DEVICE AREA
C       WHICH UTILIZES THE ENTIRE DISPLAY SURFACE, AND OUTPUT VALUES IN
C       LABEL AND DATA ARRAYS, ADJACENT TO THE PIE-CHART WHICH HAS JUST
C       BEEN GENERATED.
C
        CALL USET ('DEVICE')
        CALL USET ('PERCENTUNITS')
        CALL UDAREA (0.,100.,0.,100.)
        DO 1 I=1,NUMBER
        Y=Y-(100./FLOAT(NUMBER+1)
        CALL UMOVE (0.,Y)
        CALL UPRNT1 (LABELS(I),'TEXT')
        CALL UPRNT1 ('-;', 'TEXT')
        CALL UPRNT1 (DATA(I),'INTEGER')
        CALL UPRNT1 ('%;','TEXT)
    1   CONTINUE
C
C       WRAP-UP GRAPHICS ACTIVITY AND TERMINATE THIS FORTRAN PROGRAM.
C
        CALL UEND
        STOP
        END
```

EXAMPLE VII-6

ALGOL - 4%

COBOL - 30%

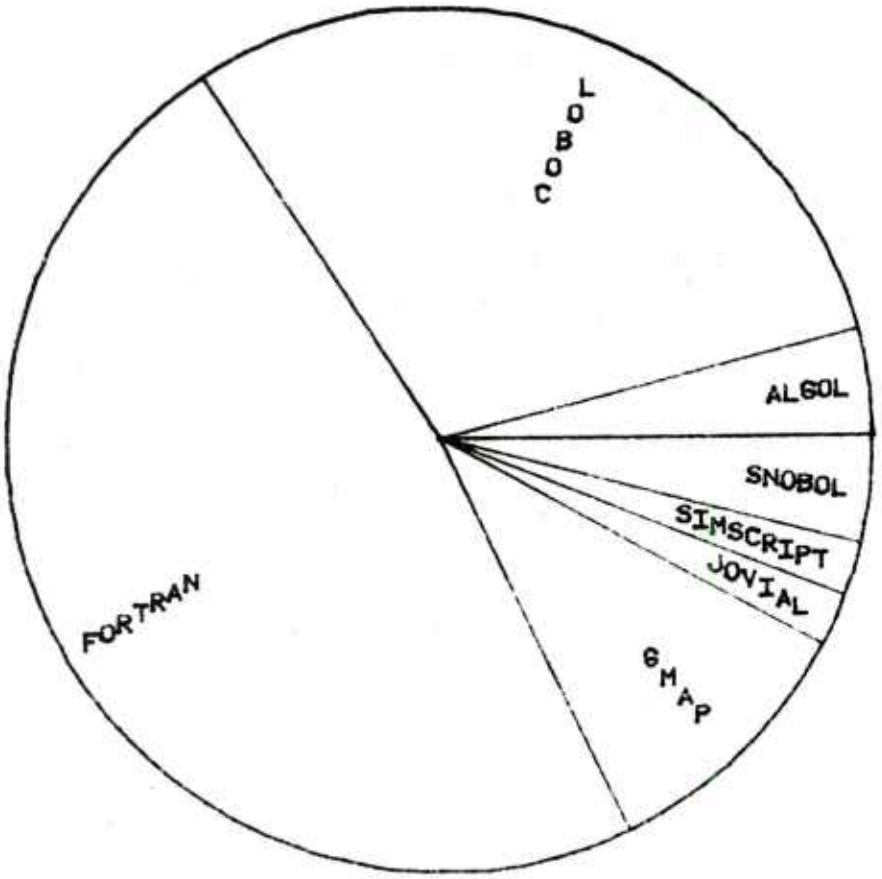
FORTRAN - 48%

GMAP - 10%

JOVIAL - 2%

SIMSCRIPT - 2%

SNOBOL - 4%



TYPICAL LANGUAGE UTILIZATION

```
      PARAMETER NUMBER=7
      DIMENSION DATA(NUMBER)
      CHARACTER LABELS*12(NUMBER)
      DATA DATA,Y/4.,30.,48.,10.,2.,2.,4.,100./
      DATA LABELS/'ALGOL;','COBOL;','FORTRAN;','GMAP;','JOVIAL;',
     &            'SIMSCRIPT;','SNOBOL;'/
      CALL USTART
      CALL UPSET ('TERMINATOR',';')
      CALL USET ('XALPHABETIC')
      CALL UPSET ('XLABEL','TYPICAL LANGUAGE UTILIZATION;')
      CALL UPIE (DATA,FLOAT(NUMBER),LABELS,12.)
      CALL USET ('DEVICE')
      CALL USET ('PERCENTUNITS')
      CALL UDAREA (0.,100.,0.,100.)
      DO 1 I = 1, NUMBER
      Y = Y - (100. / FLOAT(NUMBER+1))
      CALL UMOVE (0.,Y)
      CALL UPRNT1 (LABELS(I),'TEXT')
      CALL UPRNT1 (' - ','TEXT')
      CALL UPRNT1 (DATA(I),'INTEGER')
      CALL UPRNT1 ('%;','TEXT')
    1 CONTINUE
      CALL UEND
      STOP
      END
```

CHAPTER VIII

COORDINATE SYSTEMS AND TRANSFORMATIONS

An Overview of GCS Coordinate Systems

The GCS coordinate system options have been designed to provide adequate flexibility for the user in order to enable him to work in whatever type of coordinate system environment is most natural and convenient for him to use at a particular point in his program. There are several areas in which basic options exist; however, it is imperative that the user fully understand the basic coordinate system under GCS before attempting to alter any of the default options.

The GCS default coordinate system which is defined at the start of a GCS program (via a CALL USTART) is a simple, pre-defined Cartesian (rectangular) X-Y coordinate system. No special options such as semi-logarithmic or log-log plotting are in effect. Axes are not rotated with respect to the screen or plotbed and no special scaling options are in force. Distances are measured from the origin, not incrementally from the last point plotted. As described in Chapter III, any drawing done in this coordinate system may be viewed through an adjustable size window which can automatically clip off irrelevant or distracting parts of the picture.

A GCS programmer may specify that he wishes to use one of three coordinate systems when working in two dimensions:

Cartesian (rectangular)
Log-log or semi-logarithmic rectangular
Polar coordinates.

Points or lines to be plotted may be identified either by absolute coordinate position or relative to the last plotted point (incremental). The scale at which plotting is to occur may be user-specified in virtual space which can be projected onto the plotting device at any desired scale or it may be directly controlled in terms of dimensions of the plotting device. These dimensions may be specified in inches, centimeters, font-units (alphanumeric character heights and widths) percent-units, or raster units.

In addition to the basic or SYSTEM axis, the user may define any number of secondary USER axis systems which may be rotated, translated, and scaled differently from the SYSTEM axis. Such secondary USER axes may be defined relative to the basic SYSTEM axis or relative to a previously defined USER axis. This feature facilitates the definition of complex coordinate systems where, for example, one moves freely from coordinates based upon a location on a satellite circling the earth. Similarly, the use of simultaneously maintaining a secondary axis whose origin is a point on the curve, oriented controlled independently of one another and defined either with respect to the SYSTEM axis or cumulatively based on successive transformations on a series of different USER axes. Since the rotations and scaling in GCS are accomplished by standard matrix operations one can obtain mathematically accurate visual representations of highly abstract vector spaces such as those common to many linear programming applications.

Coordinate System Definition

In order to provide the capabilities discussed above, GCS has incorporated a very powerful secondary axis system which allows the user to define a new axis origin. The new origin can be displaced from the current origin, oriented about the new center point, and/or have a different unit length for the X and for the Y axis. The new axis is defined through subroutine UCOSYS which is called by the following sequence:

CALL UCOSYS (DX,DY,SCLX,SCLY,ANGLE)

DX and DY are used to specify (in current units of the current coordinate system) the position of the origin for the new axis. SCLX and SCLY define the scale of the new axis unit lengths with respect to their sizes in the current axis system. ANGLE is used to specify the rotation (in current angular units) of the new axis about its origin.

The key rule to be remembered is that every axis system has an origin which is coordinate location (0.,0.); all rotation occurs about this center point. The SYSTEM origin is the primary origin for the default axis system. In virtual space, the SYSTEM origin is at virtual location (0.,0.); this location may or may not be within the window defined by the user. In device space, it is a point on the display surface (usually at the lower left corner) which is at raster position (0.,0.). At any time, the user can insure that his coordinate addressing is interpreted with respect to the primary origin by making the following call to USET prior to addressing:

CALL USET ('SYSTEMAXIS')

To request that coordinate specifications be interpreted with respect to the user's current axis system, the following option is used:

CALL USET ('USERAXIS')

The default user axis coincides with the system axis. When a new user axis is created by subroutine UCOSYS, the GCS status is automatically switched to 'USERAXIS'. When UCOSYS is invoked, there are actually two USER axes created. They are either identical to each other, or one 'lags' (in a mathematical sense) behind the other by one USER axis definition. One of these USER axes is called the WORKING axis and the other is called the REFERENCE axis. The REFERENCE axis is identical to, or 'behind' the WORKING axis. Under default conditions, the REFERENCE and WORKING axes are identical to each other; furthermore, these axes initially coincide with the SYSTEM axis. When a new USER axis is defined, it is created from the REFERENCE axis and entered as the WORKING axis transform. If UCOSYS is in the cumulative mode, then the new WORKING axis definition is placed back into the REFERENCE axis transform. Thus, the REFERENCE axis may be viewed as a permanent axis, while the WORKING axis may be considered as only temporary in nature.

When multiple coordinate systems are specified by the user, the new axis is computed independently of the previous USER axis. This is the new axis which is computed from the last permanent axis (usually the SYSTEM axis) and replaces any previously defined USER axis. This feature, which is the default condition under GCS, is selected by the following call:

CALL USET ('WORKINGAXIS')

For advanced problems, the user may wish to define multiple coordinate systems that are computed on a cumulative basis. That is, the new axis is computed from the last permanent axis, and becomes the new permanent axis. This option may be specified through the following call:

CALL USET ('REFERENCEAXIS')

There exist several useful subroutines within GCS which are subsets of subroutine UCOSYS, and facilitate the definition of a secondary axis at the current beam position. Subroutine UROTAT creates a new axis rotated about an origin defined by the current beam position. Subroutine USCALE defines an axis at the current beam position having a specified X axis and Y axis scaling. Subroutine UORIGN defines a new axis at the current beam position with no change of scale or orientation. These subroutines may be invoked by the calling sequences:

```
CALL UROTAT (ANGLE)
CALL USCALE (SCLX,SCLY)
CALL UORIGN
```

ANGLE, SCLX, and SCLY are interpreted in the same manner described for subroutine UCOSYS. After the subroutines are invoked, the current beam position is associated with coordinate location (0.,0.).

In certain situations, such as the display of unrelated objects, multiple independent secondary axis transformations must be created simultaneously. In order to operate upon and modify one system, it is necessary to save the previous system and later restore it as needed. This facility is available under GCS through the use of the following subroutines:

```
CALL USVTR (ARRAY)
```

which is used to save the current transform, and

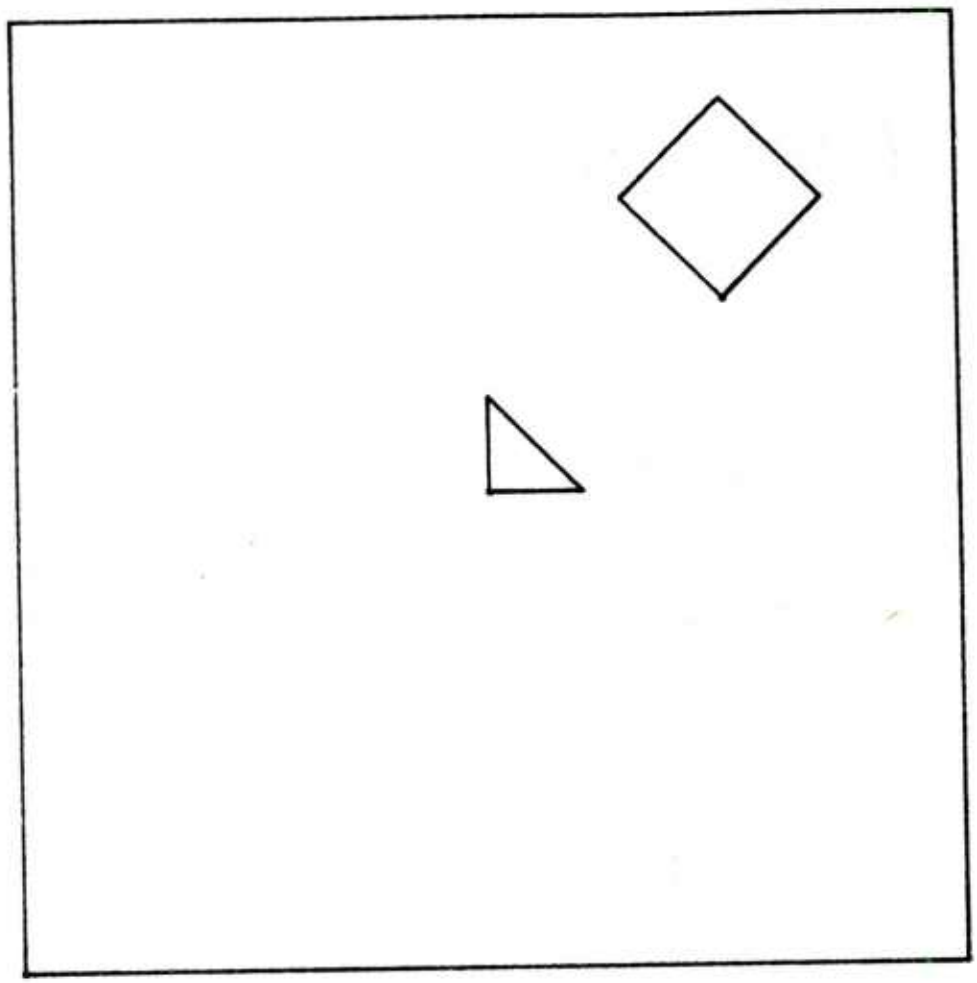
```
CALL UNSVTR (ARRAY)
```

which is called in order to restore a transformation from the array ARRAY, where ARRAY is a 21 word dimensioned variable. Note that a call to USVTR does not affect the status of the Graphics Compatibility System, but that a call to UNSVTR causes the restoration of all variables and switches to the values that they held when the companion USVTR was invoked. The user should not attempt to invoke UNSVTR for an array without previously saving a transform into that array by use of USVTR.

The secondary axis transformation feature of GCS is a very powerful mathematical system which is quite useful to the beginning as well as advanced graphics programmer. However, certain combinations of axis composition and specification will produce results which are mathematically correct but not easily understood. For example, the composition of a cumulative axis from an axis with non-uniform X and Y scaling will introduce a skew factor. For additional information on the secondary axis implementation under GCS, interested users are directed to the subroutine description section.

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE THE USE OF GCS SUBROUTINE
C     'UCOSYS'.  TWO DISPLAYS WILL BE DEFINED:  A TRIANGLE WILL BE PLOTTED
C     IN THE DEFAULT COORDINATE SYSTEM; AND A SQUARE WILL BE DRAWN IN
C     A USER-DEFINED SECONDARY AXIS SYSTEM.
C
C     ENTER GCS, DEFINE A NEW VIRTUAL WINDOW, AND OUTLINE WINDOW.
C
      CALL USTART
      CALL USET ('WORKINGAXIS')
      CALL UWINDO (-10.,10.,-10.,0.)
      CALL UOUTLN
C
C     PROVIDE PEN COMMANDS TO DRAW A TRIANGLE IN THE DEFAULT GCS
C     COORDINATE SYSTEM.
C
      CALL UMOVE (0.,0.)
      CALL UPEN (2.,0.)
      CALL UPEN (0.,2.)
      CALL UPEN (0.,0.)
C
C     DEFINE A SECONDARY AXIS WITH AN ORIGIN AT (5.,4.), HAVING THE SAME X
C     AND Y SCALE FACTORS AS THE DEFAULT SYSTEM, AND ROTATED BY 45
C     DEGREES.
C
      CALL UCOSYS (5.,4.,1.,1.,45.)
C
C     PROVIDE PEN COMMANDS TO DRAW A SQUARE WITHIN THE SECONDARY
C     AXIS THAT WE HAVE JUST DEFINED.
C
      CALL UMOVE (0.,0.)
      CALL URECT (3.,3.)
C
C     WRAP-UP GRAPHICS ACTIVITY AND TERMINATE THE FORTRAN PROGRAM.
C
      CALL UEND
      STOP
      END
```

EXAMPLE VIII-1

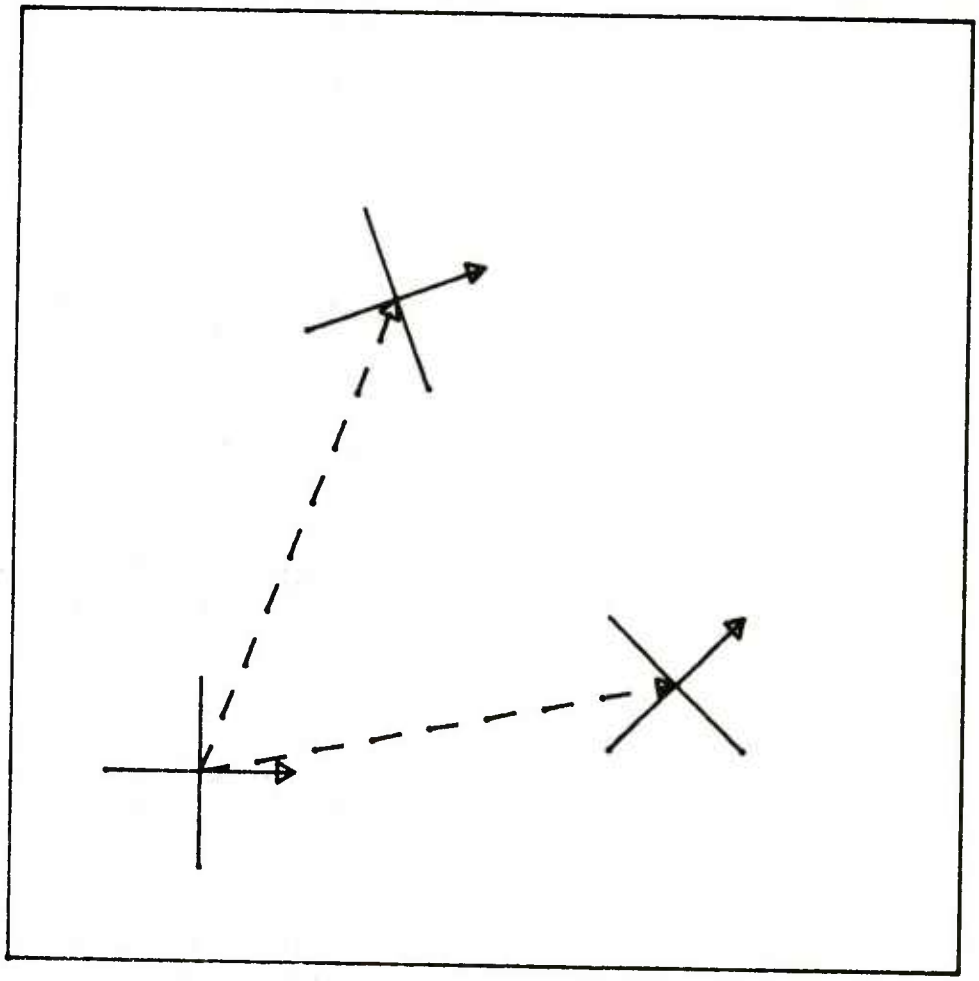


```
CALL USTART
CALL USETC'WORKINGAXIS')
CALL UWINDO (-10.,10.,-10.,10.)
CALL UOUTLN
CALL UMOVE (0.,0.)
CALL UPEN (2.,0.)
CALL UPEN (0.,2.)
CALL UPEN (0.,0.)
CALL UCOSYS (5.,4.,1.,1.,45.)
CALL UMOVE (0.,0.)
CALL URECT (3.,3.)
CALL UEND
STOP
END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE COMPOSITION OF MULTIPLE
C     USER AXES. THE AXIS DEFINITION WILL BE PERFORMED IN THE 'NON-
C     CUMULATIVE' OR 'WORKING AXIS' MODE. ENTER GCS, DEFINE A NEW
C     VIRTUAL WINDOW, AND OUTLINE DEFAULT DEVICE PLOTTING AREA.
C
      CALL USTART
      CALL USET ('WORKING AXIS')
      CALL UWINDO (-2.,8.,-2.,8.)
      CALL UOUTLN
C
C     DRAW AN AXIS AT THE CURRENT AXIS ORIGIN. THIS AXIS DISPLAY
C     SUBROUTINE WILL BE INVOKED DURING LATER PORTIONS OF THIS
C     PROGRAMMING EXAMPLE.
C
      CALL AXIS
C
C     MOVE BEAM/PEN TO THE ORIGIN OF THE CURRENT (SYSTEM) AXIS, THEN
C     DEFINE A SECONDARY AXIS WITH AN ORIGIN AT (2.,5.) AND ROTATED BY
C     TWENTY DEGREES.
C
      CALL UMOVE (0.,0.)
      CALL UCOSYS (2.,5.,1.,1.,20)
C
C     DRAW A DASHED LINE FROM THE CURRENT BEAM/PEN POSITION TO THE
C     NEW SECONDARY AXIS ORIGIN, THEN OUTLINE THE NEW AXIS.
C
      CALL UPEN1 (0.,0.,'DARROW')
      CALL AXIS
C
C     DEFINE ANOTHER AXIS WITH AN ORIGIN AT (5.,1.) AND ROTATED BY FORTY-
C     FIVE DEGREES.
C
      CALL UCOSYS (5.,1.,1.,1.,45.)
C
C     REVERT TO 'SYSTEMAXIS' MODE, AND MOVE TO THE ORIGIN OF THE
C     SYSTEM AXIS.
C
      CALL USET ('SYSTEMAXIS')
      CALL UMOVE (0.,0.)
C
C     RETURN TO 'USERAXIS' MODE, AND DRAW A DASHED LINE FROM THE ORIGIN
C     OF THE SYSTEM AXIS TO THE ORIGIN OF THE USER AXIS. AFTER THIS IS
C     DONE, OUTLINE THE CURRENT USER AXIS.
C
      CALL USET ('SYSTEMAXIS')
      CALL UMOVE (0.,0.)
C
C     RETURN TO 'USERAXIS' MODE, AND DRAW A DASHED LINE FROM THE ORIGIN
C     OF THE SYSTEM AXIS TO THE ORIGIN OF THE USER AXIS. AFTER THIS IS
C     DONE, OUTLINE THE CURRENT USER AXIS.
C
      CALL USET ('USERAXIS')
      CALL UPEN1 (0.,0.,'DARROW')
      CALL AXIS
C
C     WRAP-UP GRAPHICS ACTIVITY AND TERMINATE A FORTRAN PROGRAM.
C
      CALL UEND
      STOP
      END
```

```
C
C     SUBROUTINE TO DRAW A COORDINATE AXIS DIAGRAM AT THE CURRENT
C     ORIGIN.
C
      SUBROUTINE AXIS
      CALL UMOVE (-1.,0.)
      CALL UPEN1 (1.,0.,'LARROW')
      CALL UMOVE (0.,-1.)
      CALL UPEN (0.,1.)
      RETURN
      END
```

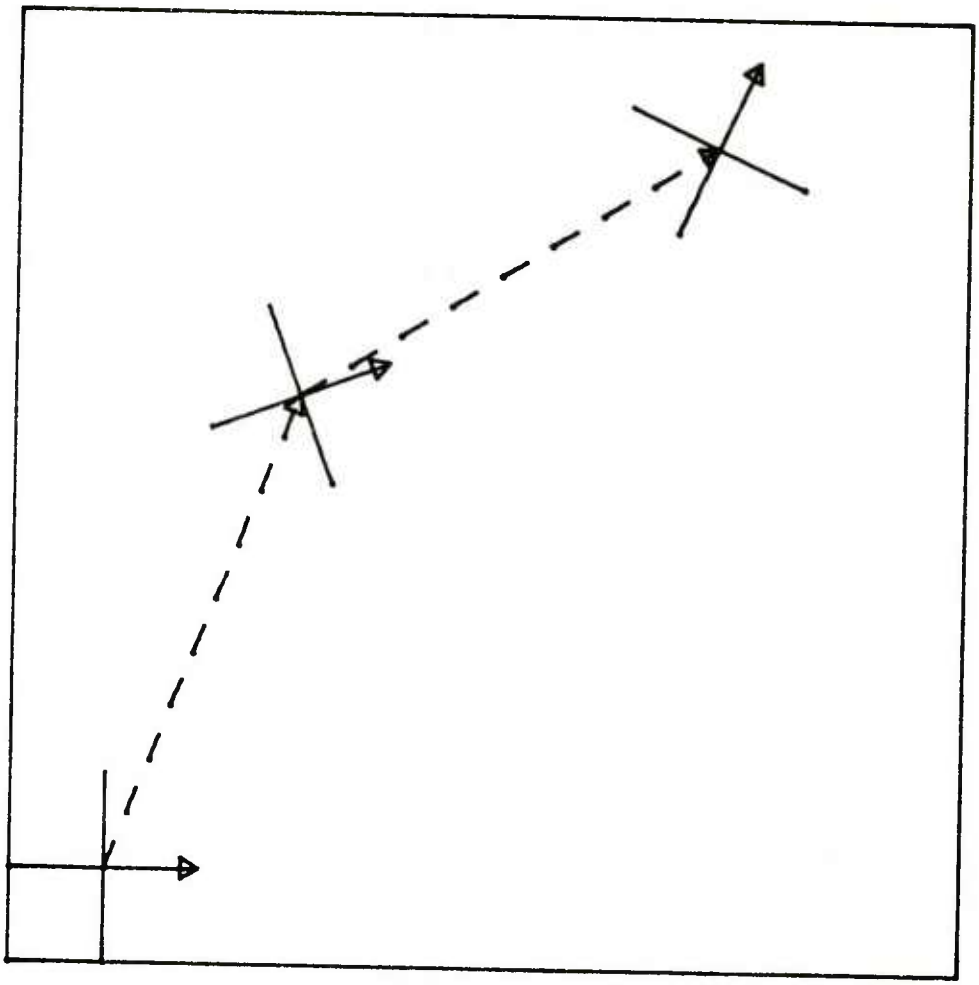
EXAMPLE VIII-2



```
CALL USTART
CALL USET ('WORKINGAXIS')
CALL UWINDO (-2.,8.,-2.,8.)
CALL UOUTLN
CALL AXIS
CALL UMOVE (0.,0.)
CALL UCOSYS (2.,5.,1.,1.,20.)
CALL UPEN1 (0.,0.,'DARROW')
CALL AXIS
CALL UCOSYS (5.,1.,1.,1.,45.)
CALL USET ('SYSTEMAXIS')
CALL UMOVE (0.,0.)
CALL USET ('USERAXIS')
CALL UPEN1 (0.,0.,'DARROW')
CALL AXIS
CALL UEND
STOP
END
SUBROUTINE AXIS
CALL UMOVE (-1.,0.)
CALL UPEN1 (1.,0.,'LARROW')
CALL UMOVE (0.,-1.)
CALL UPEN (0.,1.)
RETURN
END
```

```
C     EXAMPLE PROGRAM USED TO ILLUSTRATE COMPOSITION OF MULTIPLE
C     USER AXES.  THE AXIS DEFINITION WILL BE PERFORMED IN THE
C     'CUMULATIVE' OR 'REFERENCE AXIS' MODE.
C
C     ENTER GCS, DEFINE A NEW VIRTUAL WINDOW, AND OUTLINE DEFAULT
C     DEVICE PLOTTING AREA.
C
      CALL USTART
      CALL UWINDO (-1.,9.,-1.,9.)
      CALL UOUTLN
C
C     INDICATE THAT SECONDARY AXES ARE TO BE BUILT CUMULATIVELY.
C
      CALL USET ('REFERENCEAXIS')
C
C     DRAW AN AXIS AT THE CURRENT AXIS ORIGIN.  THIS AXIS DISPLAY ROUTINE
C     IS IDENTICAL TO THAT USED BY THE PREVIOUS EXAMPLE.
C
      CALL AXIS
C
C     MOVE BEAM/PEN TO THE ORIGIN OF THE CURRENT (SYSTEM) AXIS, THEN
C     DEFINE A SECONDARY AXIS WITH AN ORIGIN AT (2.,5.) AND ROTATED BY
C     TWENTY DEGREES.
C
      CALL UMOVE (0.,0.)
      CALL UCOSYS (2.5.,1.,1.,20.)
C
C     DRAW A DASHED LINE FROM THE CURRENT BEAM/PEN POSITION TO THE
C     NEW SECONDARY AXIS ORIGIN, THEN OUTLINE THE NEW AXIS.
C
      CALL UPEN1 (0.,0.,'DARROW')
      CALL AXIS
C
C     NOW MOVE TO THE ORIGIN OF THE CURRENT SECONDARY COORDINATE
C     SYSTEM, THEN DEFINE ANOTHER SECONDARY AXIS WITH AN ORIGIN AT
C     (5.,1.) AND ROTATED BY FORTY-FIVE DEGREES.
C
      CALL UMOVE (0.,0.)
      CALL UCOSYS (5.,1.,1.,1.,45.)
C
C     DRAW A DASHED LINE FROM THE CURRENT BEAM/PEN POSITION (THE
C     ORIGIN OF THE PREVIOUS SECONDARY AXIS) TO THE ORIGIN OF A CURRENT
C     SECONDARY AXIS, THEN OUTLINE THE CURRENT AXIS.
C
      CALL UPEN1 (0.,0.,'DARROW')
      CALL AXIS
C
C     WRAP-UP GRAPHICS ACTIVITY AND TERMINATE A FORTRAN PROGRAM.
C
      CALL UEND
      STOP
      END
C
      SUBROUTINE AXIS
      CALL UMOVE (-1.,0.)
      CALL UPEN1 (1.,0.,'LARROW')
      CALL UMOVE (0.,-1.)
      CALL UPEN (0.,1.)
      RETURN
      END
```

EXAMPLE VIII-3

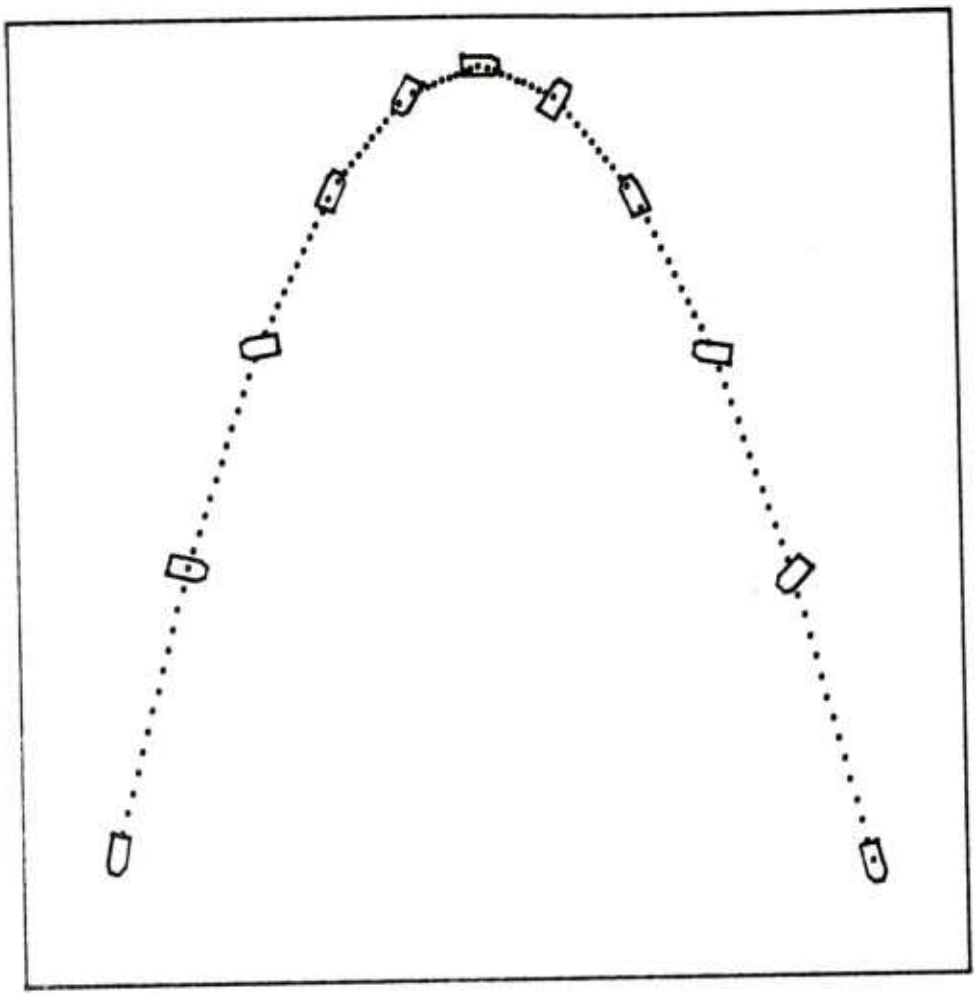


```
CALL USTART
CALL UWINDO (-1.,9.,-1.,9.)
CALL UOUTLN
CALL USET ('REFERENCEAXIS')
CALL AXIS
CALL UMOVE (0.,0.)
CALL UCOSYS (2.,5.,1.,1.,20.)
CALL UPEN1 (0.,0.,'DARROW')
CALL AXIS
CALL UMOVE (0.,0.)
CALL UCOSYS (5.,1.,1.,1.,45.)
CALL UPEN1 (0.,0.,'DARROW')
CALL AXIS
CALL UEND
STOP
END
SUBROUTINE AXIS
CALL UMOVE (-1.,0.)
CALL UPEN1 (1.,0.,'LARROW')
CALL UMOVE (0.,-1.)
CALL UPEN (0.,1.)
RETURN
END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF GCS
C     SUBROUTINE 'UCOSYS' TO DEFINE A DYNAMIC SECONDARY AXIS SYSTEM.
C     THE EXAMPLE CALCULATES THE HORIZONTAL AND VERTICAL
C     COMPONENTS OF VELOCITY AND DISPLACEMENT FOR A PROJECTILE
C     WHICH TUMBLES DURING FLIGHT. THIS EXAMPLE IS IDENTICAL TO THE
C     PREVIOUS PROGRAM WITH THE EXCEPTION OF THE ADDED ROTATIONAL
C     COMPONENT.
C
C     INITIALIZE TIME, ROTATIONAL ANGLE, VELOCITY, AND DISPLACEMENTS
C
      DATA T,THETA,VO,XO,YO/0.,75.,107.,-180.,0./
C
C     ENTER GCS, DEFINE DISPLACEMENT WINDOW, AND OUTLINE DEFAULT
C     DEVICE PLOTTING AREA.
C
      CALL USTART
      CALL USET ('WORKINGAXIS')
      CALL UWINDO (-225.,225.,-15.,95.)
      CALL UOUTLN
C
C     MOVE INITIAL DISPLACEMENT COORDINATES, SPECIFY THAT SOFTWARE
C     CHARACTERS ARE DESIRED, LINE CHARACTER SIZE, AND SET DASH SPEC.
C
      CALL UMOVE (XO,YO)
      CALL USET ('SOFTWARE')
      CALL UPSET ('HORIZONTAL', 6.)
      CALL UPSET ('VERTICAL',3.)
      CALL UPSET ('SETDASH',92.)
C
C     DEFINE LOOP TO CALCULATE HORIZONTAL AND VERTICAL
C     DISPLACEMENTS.
C
      DO 1 T=1,11
      X=(.707*VO*T)+XO
      Y=T*(.707*VO-(16.*T))+YO
C
C     SPECIFY THE SECONDARY AXIS AT COMPUTED DISPLACEMENT
C     COORDINATES THEN DRAW THE FLIGHT PATH OF THE PROJECTILE TO THIS
C     ORIGIN.
C
      CALL UCOSYS (X,Y,1.,1.,THETA)
      CALL UPEN1 (0.,0.,'DASH')
C
C     SETUP A NEW WINDOW FOR DRAWING THE PROJECTILE ITSELF WITHIN THE
C     ROTATED COORDINATE SYSTEM WE HAVE JUST DEFINED. A DISTORTED 'D'
C     REPRESENTS THE PROJECTILE.
C
      CALL UWINDO (0.,100.,0.,100.)
      CALL UWHERE (X,Y)
      CALL UPEN1 (X,Y,'DD')
C
C     REDEFINE DISPLACEMENT WINDOW, UPDATE TIME AND ROTATIONAL
C     ANGLE.
C
      CALL UWINDO (-225.,225.,-15.,95.)
      T=T+0.4758
      THETA=THETA+57.
    1 CONTINUE
```

```
C
C     WRAP-UP GRAPHICS ACTIVITY AND TERMINATE THIS FORTRAN PROGRAM.
C
      CALL UEND
      STOP
      END
```

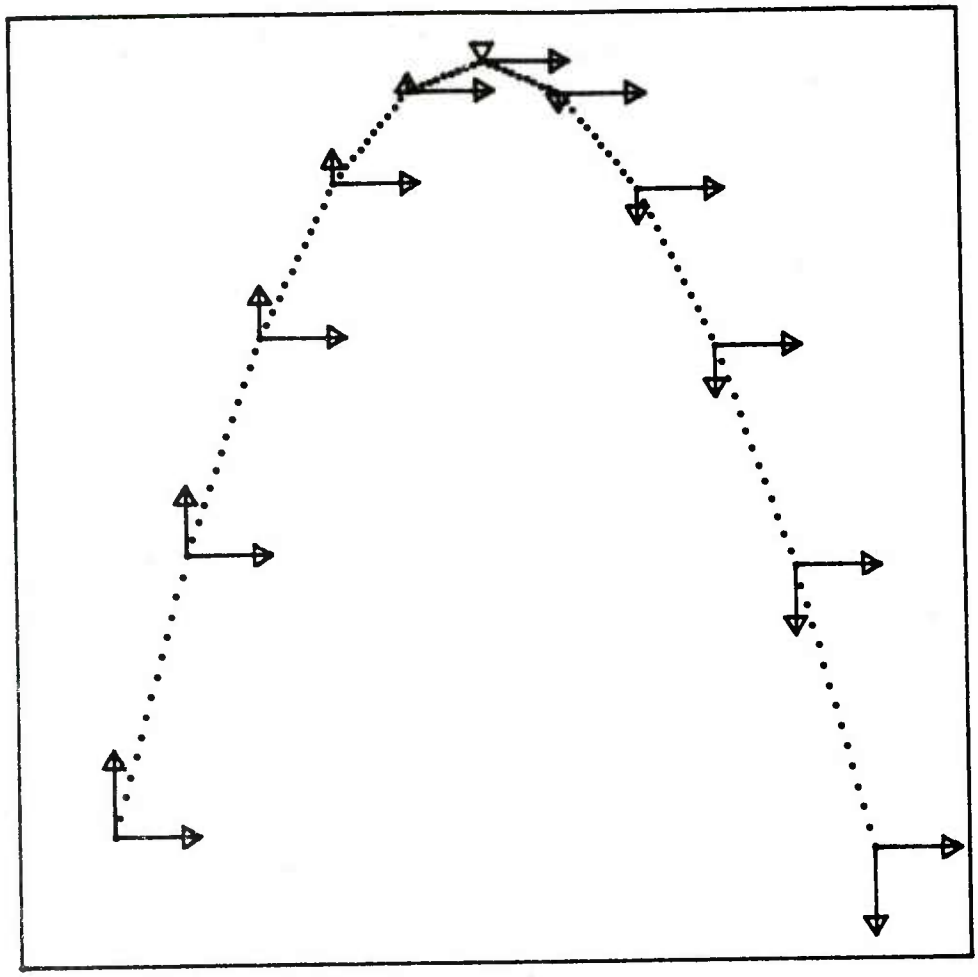
EXAMPLE VIII-4



```
      DATA T,THETA,V0,X0,Y0/0.,75.,107.,-100.,0./
      CALL USTART
      CALL USET ('WORKINGAXIS')
      CALL UWINDO (-225.,225.,-15.,95.)
      CALL UOUTLN
      CALL UMOVE (X0,Y0)
      CALL USET ('SOFTWARE')
      CALL UPSET ('HORIZONTAL',6.)
      CALL UPSET ('VERTICAL',3.)
      CALL UPSET ('SETDASH',92.)
      DO 1 I = 1, 11
      X = (.707*V0*T) + X0
      Y = T * (.707 * V0 - (16.*T)) + Y0
      CALL UCOSYS (X,Y,1.,1.,THETA)
      CALL UPEN1 (0.,0.,'DASH')
      CALL UWINDO (0.,100.,0.,100.)
      CALL UWHERE (X,Y)
      CALL UPEN1 (X,Y,'DD')
      CALL UWINDO (-225.,225.,-15.,95.)
      T = T + 0.4756
      THETA = THETA + 57.
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF GCS
C     SUBROUTINE 'UCOSYS' TO DEFINE A DYNAMIC SECONDARY AXIS SYSTEM.
C     THE EXAMPLE CALCULATES THE HORIZONTAL AND VERTICAL
C     COMPONENTS OF VELOCITY & DISPLACEMENT FOR A PROJECTILE.
C
C     DEFINE TIME, INITIAL VELOCITY, & INITIAL DISPLACEMENT LOCATION.
C
      DATA T,VO,XO,YO/0.,107.,-190.,0./
C
C     ENTER GCS, DEFINE DISPLACEMENT WINDOW, AND OUTLINE THIS WINDOW.
C
      CALL USTART
      CALL USET ('WORKINGAXIS')
      CALL UWINDO (-225,225.,-15.,95.)
      CALL UOUTLN
C
C     CALCULATE HORIZONTAL VELOCITY COMPONENT, VELOCITY SCALE
C     FACTOR, MOVE TO INITIAL DISPLACEMENT COORDINATES, AND SET DASH
C     SPECS.
C
      VX=.707*VO
      ZETA=11.*VX
      CALL UMOVE (XO,YO)
      CALL UPSET ('SETDASH',92.)
C
C     DEFINE LOOP TO CALCULATE DISPLACEMENTS, AND VERTICAL VELOCITY
C     COMPONENT.
C
      DO 1 I=1,11
      X=(VX*T)+XO
      VY=2.*(VX-32.*T)
      Y=T*(VX-(16.*T))+YO
C
C     SPECIFY THE SECONDARY AXIS AT COMPUTED DISPLACEMENT
C     COORDINATES THEN DRAW THE FLIGHT PATH OF THE PROJECTILE TO THIS
C     ORIGIN
C
      CALL UCOSYS (X,Y,1.,1.,0.)
      CALL UPEN1 (0.,0.,'DASH')
C
C     DEFINE A NEW WINDOW FOR THE VELOCITY AND PLOT THE HORIZONTAL &
C     VERTICAL COMPONENTS OF THE PROJECTILE'S VELOCITY.
C
      CALL UWINDO (0.,ZETA,-ZETA,ZETA)
      CALL USET ('RELATIVE')
      CALL UPEN1 (VX,0.,'LARROW')
      CALL UMOVE (-VX,0.)
      CALL UPEN1 (0.,VY,'LARROW')
      CALL UMOVE (0.,-VY)
C
C     REDEFINE DISPLACEMENT WINDOW, UPDATE TIME, AND CONTINUE A LOOP.
C
      CALL USET ('ABSOLUTE')
      CALL UWINDO (-225.,225.,-15.,95.)
      T=T+.4758
    1 CONTINUE
C
C     WRAP-UP GRAPHICS ACTIVITY AND TERMINATE THIS FORTRAN PROGRAM.
C
      CALL UEND
      STOP
      END
```

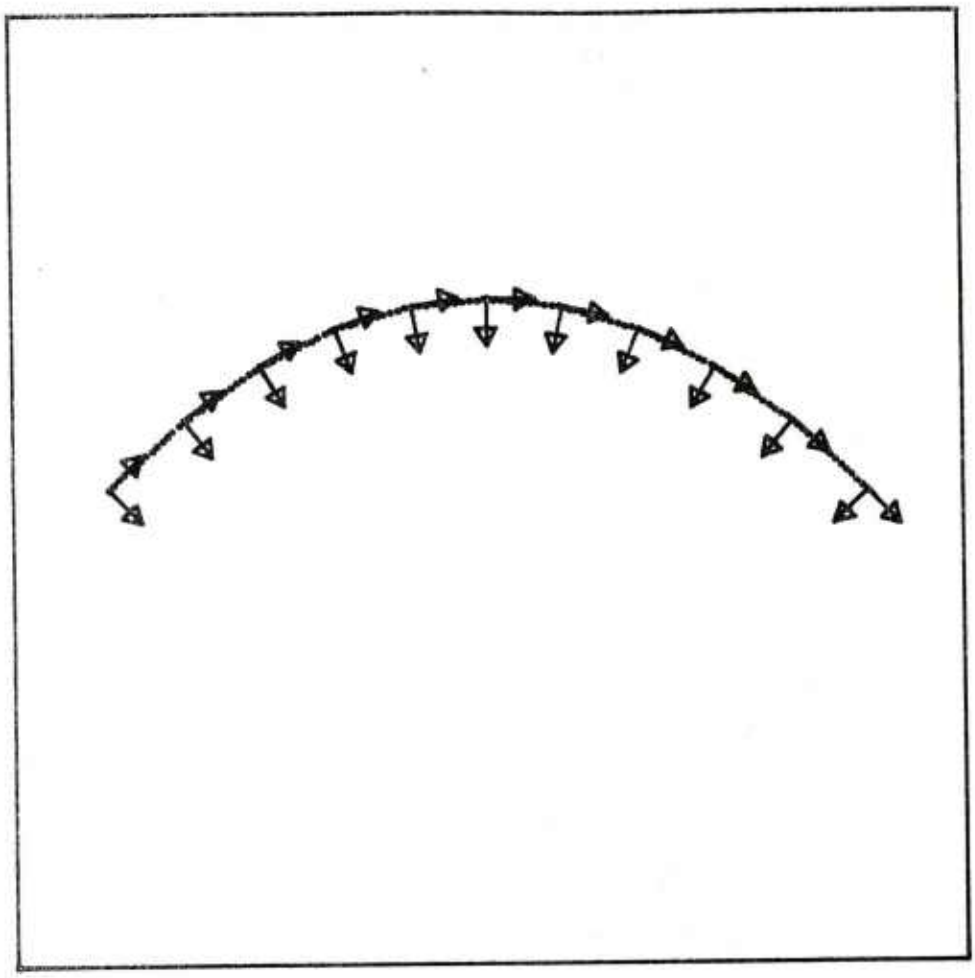
EXAMPLE VIII-5



```
      DATA T,V0,X0,Y0/0.,107.,-168.,0./
      CALL USTART
      CALL USET ('WORKINGAXIS')
      CALL UWINDO (-225.,225.,-15.,95.)
      CALL UOUTLN
      VX = .707 * V0
      ZETA = 11. * VX
      CALL UMOVE (X0,Y0)
      CALL UPSET ('SETDASH',92.)
      DO 1 I = 1, 11
      X = (VX*T) + X0
      VY = 2. * (VX - (32.*T))
      Y = T * (VX - (16.*T)) + Y0
      CALL UCOSYS (X,Y,1.,1.,0.)
      CALL UPEN1 (0.,0.,'DASH')
      CALL UWINDO (0.,ZETA,-ZETA,ZETA)
      CALL USET ('RELATIVE')
      CALL UPEN1 (VX,0.,'LARROW')
      CALL UMOVE (-VX,0.)
      CALL UPEN1 (0.,VY,'LARROW')
      CALL UMOVE (0.,-VY)
      CALL USET ('ABSOLUTE')
      CALL UWINDO (-225.,225.,-15.,95.)
      T = T + .4758
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE APPLICATION OF GCS
C       SUBROUTINE UCOSYS TO DEFINE A DYNAMIC SECONDARY AXIS SYSTEM.
C       THE EXAMPLE DISPLAYS THE TANGENT AND NORMAL COMPONENTS OF
C       TRAJECTORY AND DISPLACEMENT FOR A PROJECTILE.
C
C       DEFINE TIME INITIAL VELOCITY AND INITIAL DISPLACEMENT LOCATION.
C
        DATA T,VO,XO,YO/0.,107.,-190.,0./
C
C       ENTER GCS DEFINE DISPLACEMENT WINDOW, AND OUTLINE THIS WINDOW.
C
        CALL USTART
        CALL USET ('WORKINGAXIS')
        CALL UWINDO (-225.,225.,-225.,225.)
        CALL UOUTLN
C
C       CALCULATE TANGENT VELOCITY COMPONENT MOVE TO INITIAL
C       DISPLACEMENT COORDINATES, AND SET DASH SPECS.
C
        VX=.707*VO
        CALL UMOVE (XO,YO)
        CALL UPSET ('SETDASH',92.)
        CALL USET ('RADIANS')
C
C       DEFINE LOOP TO CALCULATE DISPLACEMENTS
C
        DO 1 I=1,11
        X=(VX*T)+XO
        Y=T*(VX-16.*T))+YO
        DYDX=1-(32*(X-XO)/VX**2)
C
C       SPECIFY THE SECONDARY AXIS AT COMPUTED DISPLACEMENT
C       COORDINATES AND ROTATED TO THE SLOPE OF THE TANGENT.  THEN DRAW
C       THE FLIGHT PATH OF THE PROJECTILE TO THIS ORIGIN.
C
        CALL UCOSYS (X,Y,1.,1.,ATAN(DYDX))
        CALL UPEN1 (0.,0.,'DASH')
C
C       DEFINE A NEW WINDOW AND PLOT THE TANGENT & NORMAL COMPONENTS
C       OF THE PROJECTILE'S TRAJECTORY.
C
        CALL UWINDO (0.,1.,-1.,0.)
        CALL USET ('RELATIVE')
        CALL UPEN1 (.05,0.,'LARROW')
        CALL UMOVE (-.05,0.)
        CALL UPEN1 (0.,-.05,'LARROW')
        CALL UMOVE (0.,.05)
C
C       REDEFINE DISPLACEMENT WINDOW, UPDATE TIME, AND CONTINUE A LOOP.
C
        CALL USET ('ABSOLUTE')
        CALL UWINDO (-225.,225.,-225.,225.)
        T=T+.4758
    1   CONTINUE
C
C       WRAP-UP GRAPHICS ACTIVITY AND TERMINATE THIS FORTRAN PROGRAM.
C
        CALL UEND
        STOP
        END
```

EXAMPLE VII-6



```
      DATA T,V0,X0,Y0/0.,107.,-160.,0./
      CALL USTART
      CALL USET ('WORKINGAXIS')
      CALL UWINDO (-225.,225.,-225.,225.)
      CALL UOUTLN
      VX = .707 * V0
      CALL UMOVE (X0,Y0)
      CALL UPSET ('SETDASH',92.)
      CALL USET ('RADIANS')
      DO 1 I = 1, 11
      X = (VX*T) + X0
      Y = T * (VX - (16.*T)) + Y0
      DYDX = 1 - (32 * (X-X0) / VX**2)
      CALL UCOSYS (X,Y,1.,1.,ATAN(DYDX))
      CALL UPEN1 (0.,0.,'DASH')
      CALL UWINDO (0.,1.,-1.,0.)
      CALL USET ('RELATIVE')
      CALL UPEN1 (.05,0.,'LARROW')
      CALL UMOVE (-.05,0.)
      CALL UPEN1 (0.,-.05,'LARROW')
      CALL UMOVE (0.,.05)
      CALL USET ('ABSOLUTE')
      CALL UWINDO (-225.,225.,-225.,225.)
      T = T + .4758
    1 CONTINUE
      CALL UEND
      STOP
      END
```

CHAPTER IX

THREE DIMENSIONAL GRAPHICS

The three dimensional version of GCS gives to the user the ability to describing his environment in three dimensions. Each of the primary two dimensional GCS routines have a three dimensional counterpart: U3PEN, U3MOVE, U3DRAW, U3PRNT, U3WRIT, U3CSYS, and U3WHER. The coordinates used by each of these routines may be specified as (X,Y,Z) rectangular coordinates, (R,O,Z) cylindrical coordinates, or (R,O,ι) spherical coordinates. Use of the 3-D routines does not preclude use of the 2-D routines; the 2-D routines are defined so as to draw parallel to the current X-Y plane on a plane specified by a default Z coordinate, contained in the Graphics Status Area (GSA). This Z value may be set by the user with an UPSET ('ZVALUE', value) call. The default ZVALUE is zero. By using both 2-D and 3-D routines an image may be described in 3-space.

In order to specify the appearance of the object being drawn, the location of the viewer and the direction in which he is looking must be specified. By calling UVIEW (XVIEW,YVIEW,ZVIEW,XSITE,YSITE,ZSITE), the user can specify where the view point is located and at what point in the evironment he is looking (view site). The environment will then be displayed from this viewpoint with the view site located on a line which passes through the center of the viewport. The viewport indicates the boundaries of the user's field of view and also how far the viewport is from the view point or the viewsite. The X-coordinates of the viewport will range in the interval (-A/2,A/2) and the Y-coordinates in the interval (-B/2,B/2). Default values for UVIEW are (0.,0.,150.,0.,0.,0.) and for UVWPRT are (200.,200.,0.). GCS will also automatically clip points outside of the viewing pyramid (behind the view point or outside the viewport). The user can select whether the resulting image will be mapped to the screen with a PERSPECTIVE (default) or ORTHOGONAL projection.

The user may specify which portion of this viewport will be displayed on the screen by setting his UWINDO. The UWINDO boundaries may or may not overlap the viewport. If no portion of the UWINDO corresponds to the viewport, the viewport will be expanded to encompass the UWINDO if the UWINDO changes. If the viewport is specified smaller than the UWINDO, the UWINDO will contract to encompass the viewport.

As in 2-D GCS, the user may construct arbitrary user coordinate systems using the calls U3CSYS, U3SCAL, and U3ROTA. These work similar to the 2-D case with one exception. Since 3-D rotations are not commutative, the order of rotation must be specified with a suitable USET call: 'XYZ', 'XZY', 'YXZ', 'ZXY,' or 'ZYX'. The default sequence is **'ZYX'**.

A call to the 2-D routine UCOSYS will cause a rotation about the Z axis. Any rotation around other than the system Z axis will cause the XY plane to cease being parallel to the screen and will effect the output of symbols and curves (see below).

Curve generation using the UARC, UCRCLE, UCONIC, UPLYGN, and URECT routines function as they did before. However, since they are 2-D routines, their output will be created in the XY plane specified by the current ZVALUE parameter as indicated above. In addition, software characters generated by U3PRNT, U3WRIT, and UAOUT will also appear in the XY plane. Hardware characters still appear on the plane of the display surface. Line terminators (arrowheads and software characters) and tic marks appear on a plane specified by the two end points of the line and a third point which the user can specify via the UPOINT(X,Y,Z) subroutine. The default UPOINT is a point on the default XY-plane outside the default UWINDO boundaries. This point is (-.6931471806, -1.096122887, 0.).

Textual output for three dimensions has been expanded slightly to allow another text output mode. This mode is 'XYZCOORDINATES' which will print the three components

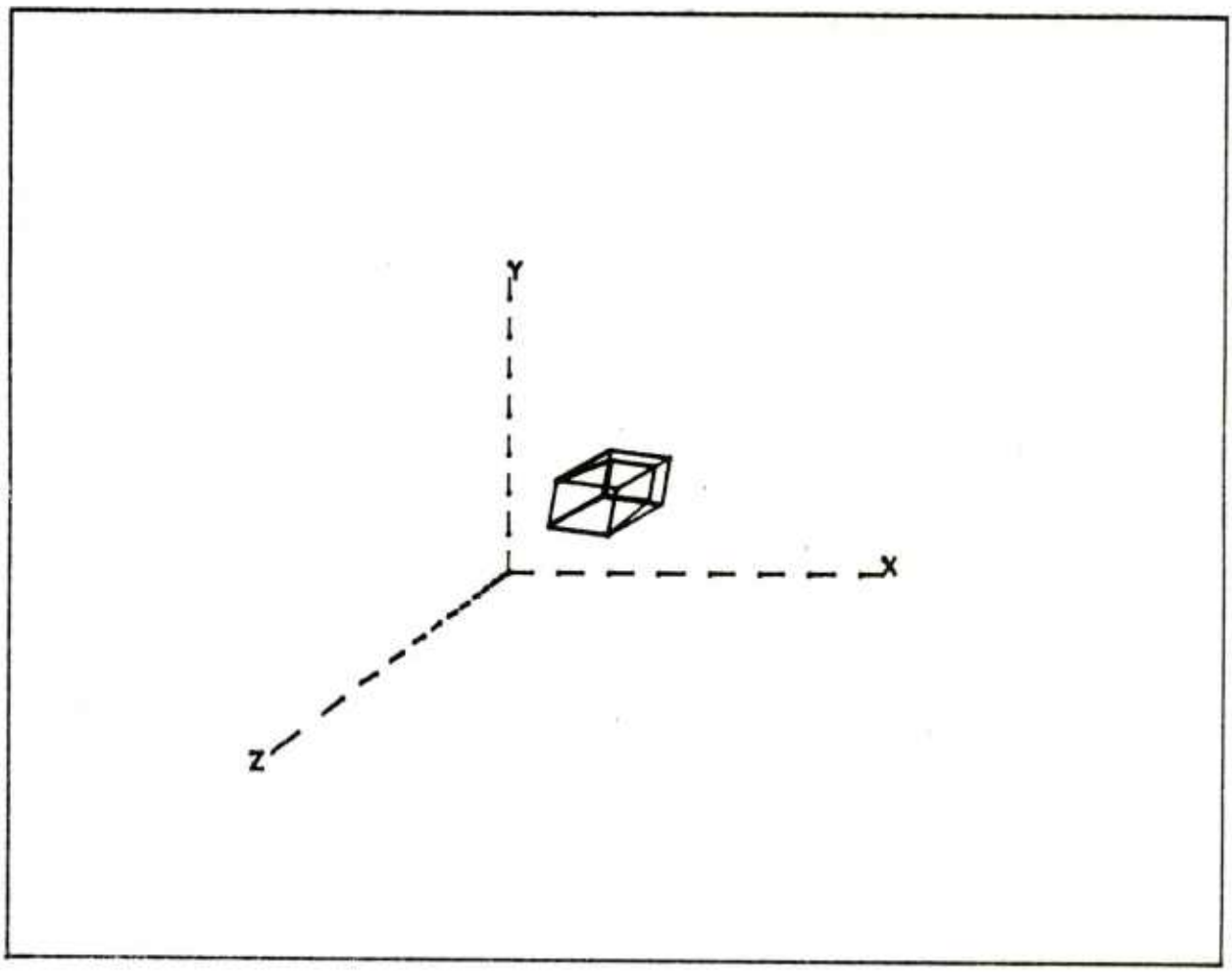
of a set of 3-D coordinates. 'XYCOORDINATES' are still available for 2-D coordinates. All other text functions of 2-D GCS are processed as before. Since the text created by U3PRNT and U3WRIT is displayed in the current user X-Y plane, specified by the Z-coordinate of the specified location, suitable choices of UVIEW will result in text which when displayed will be backwards. This is a result of viewing the text from behind it.

Drawing 3-D images is primarily intended for use in VIRTUAL space. However, a user who wishes to do so can directly address the face of the display surface in DEVICE space. If this is done, no clipping against viewport boundaries takes place. The requested graphical data is displayed projected orthogonally onto the device screen. For most devices, the Z-value will be ignored and the X,Y components will be used as screen addresses after conversion to device units.

To facilitate displaying different sections of the viewport, an additional window and display area routine has been implemented. ULOOK lets the user specify a portion of the display surface (UDAREA) in device units. The window (UWINDO) dimensions are then adjusted to move the UWINDO to display the equivalent portion of virtual space (viewport). The effect is that of moving a template around on the viewport.

```
C     THIS PROGRAM DEMONSTRATES THE USE OF SEVERAL 3-D SUBROUTINES
C     AND THE DIFFERENCE BETWEEN ORTHOGONAL AND PERSEPCTIVE
C     PLOTTING.
C
      CALL USTART
      CALL UPSET ('SETD',1.)
      CALL UPSET ('TERMINATOR', ';')
      CALL USET ('PERC')
      CALL UDAREA (0.,100.,0.,100.)
      CALL AXIS
      CALL USET ('ORTH')
      CALL BOX
      CALL USET ('PERC')
      CALL UDAREA (0.,100.,0.,100.)
      CALL UWINDO (-100.,100.,-100.,100.)
      CALL USET ('PERSPECTIVE')
      CALL BOX
      CALL UEND
      STOP
      END
      SUBROUTINE AXIS
      CALL UOUTLN
      CALL USET ('DASH')
      CALL U3MOVE (-20.,-20.,-20.)
      CALL U3PEN (50.,-20.,-20.)
      CALL UPRNT1 ('X;', 'TEXT')
      CALL U3MOVE (-20.,-20.,-20.)
      CALL U3PEN (-20.,50.,-20.)
      CALL UPRNT1 ('Y;', 'TEXT')
      CALL U3MOVE (-20.,-20.,-20.)
      CALL U3PEN (-20.,-20.,100.)
      CALL UPRNT1 ('Z;', 'TEXT')
      CALL USET ('LINE')
      CALL U3CSYS (25.,25.,25.,1.,1.,1.,10.,10.,0.)
      CALL UVIEW (-20.,-20.,-150.,0.,0.,0.)
      RETURN
      END
      SUBROUTINE BOX
      CALL U3MOVE (0.,0.,0.)
      CALL U3PEN (10.,0.,0.)
      CALL U3PEN (10.,10.,0.)
      CALL U3PEN (0.,10.,0.)
      CALL U3PEN 0.,0.,0.)
      CALL U3MOVE (10.,0.,0.)
      CALL U3PEN (10.,0.,60.)
      CALL U3PEN (10.,10.,60.)
      CALL U3PEN (10.,10.,0.)
      CALL U3MOVE (0.,10.,0.)
      CALL U3PEN (0.,10.,60.)
      CALL U3PEN (10.,10.,60.)
      CALL U3MOVE (0.,0.,0.)
      CALL U3PEN (0.,0.,60.)
      CALL U3PEN (0.,10.,60.)
      CALL U3PEN (0.,10.,0.)
      CALL U3MOVE (0.,0.,60.)
      CALL U3PEN (10.,0.,60.)
      RETURN
      END
```

EXAMPLE IX-1




```
CALL USTART
CALL UPSET ('TERMINATOR',';')
CALL UPSET ('SPEED',120.)
CALL USET ('PERCENTUNITS')
CALL UDAREA (0.,100.,0.,100.)
CALL UWINDO (-100.,100.,-100.,100.)
CALL UERASE
CALL AXIS
CALL USET ('ORTHOGONAL')
CALL BOX
CALL UDAREA (0.,100.,0.,100.)
CALL UWINDO (-100.,100.,-100.,100.)
CALL USET ('PERSPECTIVE')
CALL BOX
CALL UEND
STOP
END
```

```
SUBROUTINE AXIS
CALL UOUTLN
CALL USET ('DASH')
CALL U3MOVE (-20.,-20.,-20.)
CALL U3PEN (50.,-20.,-20.)
CALL UPRNT1 ('X,','TEXT')
CALL U3MOVE (-20.,-20.,-20.)
CALL U3PEN (-20.,50.,-20.)
CALL UPRNT1 ('Y,','TEXT')
CALL U3MOVE (-20.,-20.,-20.)
CALL U3PEN (-20.,-20.,100.)
CALL UPRNT1 ('Z,','TEXT')
CALL USET ('LINE')
CALL U3CSYS (25.,25.,25.,1.,1.,1.,10.,10.,0.)
CALL UVIEW (-20.,-20.,-150.,0.,0.,0.)
RETURN
END
```

```
SUBROUTINE BOX
CALL U3MOVE (0.,0.,0.)
CALL U3PEN (10.,0.,0.)
CALL U3PEN (10.,10.,0.)
CALL U3PEN (0.,10.,0.)
CALL U3PEN (0.,0.,0.)
CALL U3MOVE (10.,0.,0.)
CALL U3PEN (10.,0.,60.)
CALL U3PEN (10.,10.,60.)
CALL U3PEN (10.,10.,0.)
CALL U3MOVE (0.,10.,0.)
CALL U3PEN (0.,10.,60.)
CALL U3PEN (10.,10.,60.)
CALL U3MOVE (0.,0.,0.)
CALL U3PEN (0.,0.,60.)
CALL U3PEN (0.,10.,60.)
CALL U3PEN (0.,10.,0.)
CALL U3MOVE (0.,0.,60.)
CALL U3PEN (10.,0.,60.)
RETURN
END
```

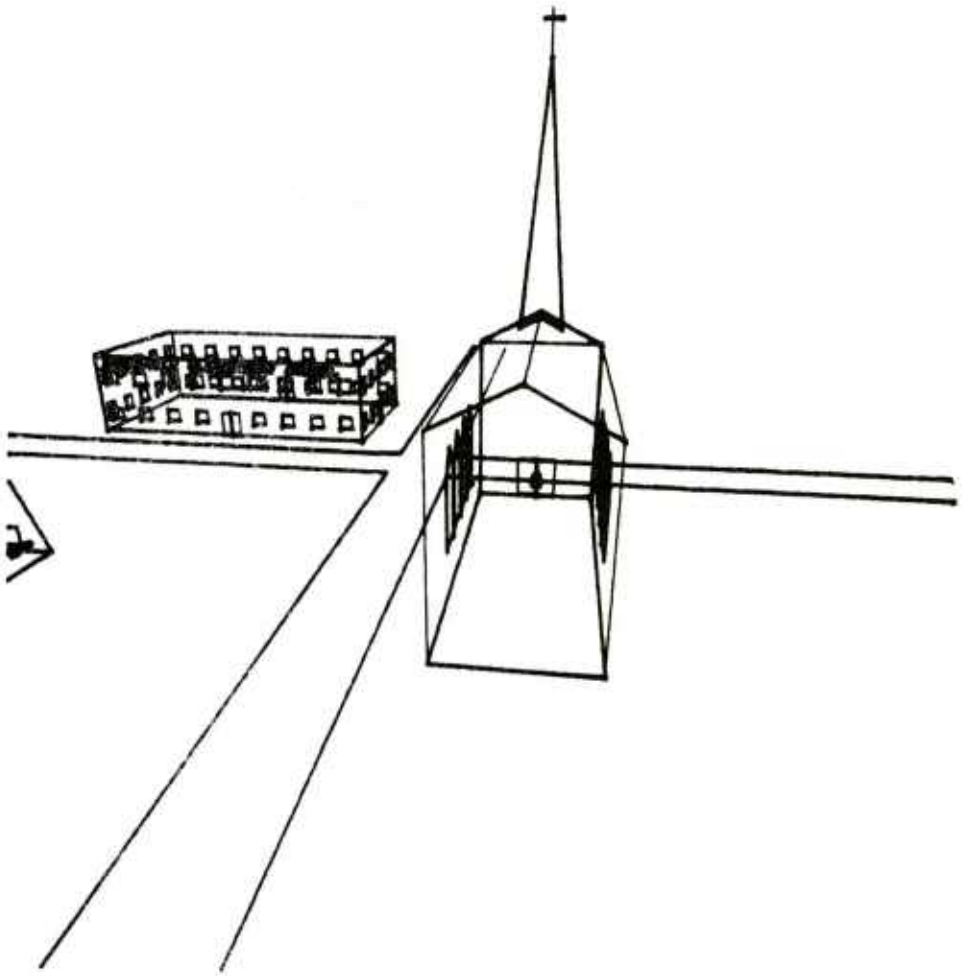
```
C       THIS PROGRAM DEMONSTRATES THE USE OF SEVERAL 3-D SUBROUTINES
C       TO PRODUCE A VILLAGE ON A TEKTRONIX 4014/4015 TERMINAL.
C
        CALL USTART
        CALL USET ('VIEW')
        CALL UVWPRT (150.)
        CALL UPSET ('TERM', '<')
        CALL UWINDO (-100.,100.,-100.,100.)
        CALL UDAREA (0.,7.,0.,7.)
        CALL UVIEW (-40.,200.,700.,-20.,20.,0.)
        CALL VILLAG
        CALL UDAREA (7.1,14.1,0.,7.)
        CALL UVIEW (-70.,-150.,50.,0.,0.,10.)
        CALL VILLAG
        CALL UEND
        STOP
        END
        SUBROUTINE VILLAG
        CALL USET ('XYZ')
        CALL USET ('SYST')
        CALL USET ('REFE')
        CALL U3CSYS (-50.,20.,0.,1.,1.,1.,90.,0.,0.)
        CALL USET ('BLACK')
        CALL CHURCH
        CALL USET ('SYST')
        CALL U3CSYS (24.,-19.,0.,1.,1.,1.,80.,-90.,0.)
        CALL USET ('RED')
        CALL SCHOOL
        CALL USET ('SYST')
        CALL U3CSYS (70.,70.,0.,0.7.0.7,0.7,0.,0.,0.)
        CALL USET ('BLUE')
        CALL PIZZA
        CALL USET ('SYST')
        CALL U3CSYS (0.,0.,0.,1.,1.,1.,0.,0.,0.)
        CALL USET ('BLACK')
        CALL ROAD
        CALL USET ('SYST')
        RETURN
        END
        SUBROUTINE SCHOOL
        CALL USET ('SOFT')
        CALL USET ('REFE')
        CALL U3MOVE (0.,0.,0.)
        CALL URECT (50.,25.)
        CALL U3MOVE 3.,4.,0.)
        CALL URECT (7.,8.)
        CALL U3MOVE (13.,4.,0.)
        CALL URECT (17.,8.)
        CALL U3MOVE (47.,4.,0.)
        CALL URECT (43.,8.)
        CALL U3MOVE (37.,4.,0.)
        CALL URECT (33.,8.)
        CALL U3MOVE (25.,0.,0.)
        CALL URECT (22.,8.)
        CALL URECT (28.,8.)
        DO 10 I=3,43,10
        X=I
        CALL U3MOVE (X,15.,0.)
```

```
10  CALL URECT (X+4.,20.)
    CALL USET ('WORK')
    CALL U3CSYS (50.,0.,0.,1.,1.,1.,0.,90.,0.)
    ISW=1
15  CALL U3MOVE (0.,0.,0.)
    CALL URECT (90.,25.)
    DO 20 I=3,83,10
    X=I
    CALL U3MOVE (X,16.,0.)
    CALL URECT (X+4.,20.)
    IF(I.EQ.43) GO TO 20)
    CALL U3MOVE (X,4.,0.)
    CALL URECT (X+4.,8.)
20  CONTINUE
    CALL U3MOVE (45.,0.,0.)
    CALL URECT (42.,8.)
    CALL URECT (48.,8.)
    IF ISW.EQ.2) GO TO 30
    ISW=2
    CALL U3CSYS (0.,0.,-90/,1.,1.,1.,0.,-90.,0.)
    GO TO 15
30  CALL UPSET ('VERT',3.)
    CALL UPSET ('HORI',3.)
    CALL UPRINT (0.,21.,5.'ROOSEVELT ELEMENTARY SCHOOL<')
    CALL USET ('REFE')
    CALL UPSET ('ZVAL',-90.)
    CALL U3MOVE (0.,0.,-90.)
    CALL URECT (50.,25.)
    DO 40 I=3,43,10
    X=I
    CALL U3MOVE (X-4.,-80.)
    CALL URECT (X+4.,8.)
    CALL U3MOVE (X,16.,-90.)
40  CALL URECT (X+4.,20.)
    CALL UPSET ('ZVAL',0.)
    RETURN
    END
    SUBROUTINE CHURCH
    CALL USET ('REFE')
    CALL U3MOVE (0.,0.,0.)
    CALL URECT (30.,40.)
    CALL U3MOVE (15.,0.,0.)
    CALL URECT (10.,10.)
    CALL URECT (20.,10.)
    CALL U3MOVE (14.,3.,0.)
    CALL URECT (14.25,6.)
    CALL U3MOVE (15.,75,3.,0.)
    CALL URECT (16.,6)
    CALL U3MOVE (0.,40.,0.)
    CALL U3PNE (30.,40.,0.)
    CALL USET ('WORK')
    CALL U3CSYS (30.,0.,0.,1.,1.,1.,0.,90.,0.)
    ISW=1
10  CALL U3MOVE (0.,0.,0.)
    CALL URECT (80.,40.)
    DO 20 I=20,40,10
    X=I
    CALL U3MOVE (X,10.,0.)
```

```
      CALL U3PEN (X,25.,0.)
      CALL U3PEN (X+4.,30.,0.)
      CALL U3PEN (X+8.,25.,0.)
      CALL U3PEN (X+8.,10.,0.)
20    CALL U3PEN (X,10.,0.)
      IF (ISW,EQ.2) GO TO 30
      ISW=2
      CALL U3CSYS (0.,0.,0.,1.,1.,1.,0.,90.,0.)
      GO TO 10
30    CALL USET ('REFE')
      CALL U3MOVE (0.,40.,-80.)
      CALL U3PEN (15.,47.5,-80.)
      CALL U3PEN (30.,40.,-80.)
      CALL U3MOVE (0.,0.,-80.)
      CALL E3PEN (30.,0.,-80.)
      CALL U3MOVE (15.,47.,5,-80.)
      CALL U3PEN (15.,47.5,0.)
      CALL U3MOVE (10.,45.,0.)
      CALL U3PEN (10.,45.,-10.)
      CALL U3PEN (15.,47.,5,-10.)
      CALL U3PEN (20.,45.,-10.)
      CALL U3PEN (20.,45.,0.)
      CALL U3PEN (15.,100.,-5.)
      CALL U3PEN (10.,45.,0.)
      CALL U3MOVE (10.,45.,-10.)
      CALL U3PEN (15.,100.,-5.)
      CALL U3PEN (20.,45.,-10.)
      CALL U3MOVE (15.,100.,-5.)
      CALL U3PEN (15.,110.,-5.)
      CALL U3MOVE (13.,107.,-5.)
      CALL U3PEN (17.,107.,-5.)
      RETURN
      END
      SUBROUTINE ROAD
      CALL U3MOVE (-200.,9.,0.)
      CALL U3PEN (-9.,9.,0.)
      CALL U3PEN (-9.,200.,0.)
      CALL U3MOVE (200.,9.,0.)
      CALL U3PEN (9.,9.,0.)
      CALL U3PEN (9.,200.,0.)
      CALL U3MOVE (-200.,-9.,0.)
      CALL U3PEN (-9.,-9.,0.)
      CALL U3PEN (-9.,-200.,0.)
      CALL UMOVE (9.,-200.,0.)
      CALL U3PEN (9.,-9.,0.)
      CALL U3PEN (200.,-9.,0.)
      RETURN
      END
      SUBROUTINE PIZZA
      CALL USET ('SOFT')
      CALL USET ('REFE')
      CALL U3MOVE (0.,0.,0.)
      CALL U3CSYS (0.,0.,0.,1.,1.,1.,90.,0.,0.)
      CALL U3PEN (40.,0.,0.)
      CALL U3PEN (20.,50.,0.)
      CALL U3PEN (0.,0.,0.)
      CALL U3MOVE (20.,0.,0.)
      CALL URECT (17.5,8.)
```

```
CALL URECT (22.5,8.)
CALL U3MOVE (10.,2.,0.)
CALL URECT (15.,8.)
CALL U3MOVE (30.,2.,0.)
CALL URECT (25.,8.)
CALL U3MOVE (13.,10.,0.)
CALL URECT (27.,15.)
CALL UPSET ('HORI',2.)
CALL UPSET ('VERT', 4.)
CALL USET ('ITAL')
CALL UPRINT (15.65,11.2,'PIZZA<')
CALL USET ('GOTH')
CALL UPSET ('VERT',.4)
CALL UPSET ('HORI',.3)
CALL UPRINT (20.95,5.,'IN<')
CALL USET ('WORK')
CALL U3CSYS (40.,0.,0.,1.,1.,1.,0.,-180.,0.)
CALL UPRINT (20.65,5., 'OUT<')
CALL USET ('REFE')
CALL U3MOVE (0.,0.,-60.)
CALL U3PEN (40.,0.,-60.)
CALL U3PEN (20.,50.,-60.)
CALL U3PEN (0.,0.,-60.)
CALL U3MOVE (18.5,0.,-60.)
CALL UPSET ('ZVAL', -60.)
CALL URECT (21.5,8.)
CALL UPSET ('ZVAL',0.)
CALL USET ('REFE')
CALL U3MOVE (0.,0.,0.)
CALL U3PEN (0.,0.,-60.)
CALL U3MOVE (40.,0.,-60.)
CALL U3PEN (40.,0.,0.)
CALL U3MOVE (20.,50.,0.)
CALL U3PEN (20.,50.,-60.)
CALL USET ('WORK')
CALL USET ('YXZ')
CALL U3CSYS (0.,0.,0.,1.,1.,1.,-ATAN2(50.,20.)*57.29577+90.,90.,0.)
CALL USET ('XYZ')
CALL UPSET ('HORI',2.)
CALL UPSET ('VERT',2.5)
CALL UPRINT (5.,5.,'GIANT PIZZA $8<')
CALL USET ('REFE')
RETURN
END
```

EXAMPLE IX-2



```
CALL USTART
CALL USET ('VIEWDISTANCE')
CALL UVWPRT (150.)
CALL UPSET ('TERMINATOR',',')
CALL UWINDO (-100.,100.,-100.,100.)
CALL UDAREA (0.,10.8,0.,10.8)
CALL UVIEW (-40.,200.,70.,-20.,20.,0.)
CALL VILLAG
CALL UEND
STOP
END
```

```
SUBROUTINE VILLAG
CALL USET ('XYZ ')
CALL USET ('SYSTEMAXIS')
CALL USET ('REFERENCEAXIS')
CALL U3CSYS (-50.,20.,0.,1.,1.,1.,90.,0.,0.)
CALL USET ('BLACK')
CALL CHURCH
CALL USET ('SYSTEMAXIS')
CALL U3CSYS (24.,-10.,0.,1.,1.,1.,90.,-90.,0.)
CALL USET ('RED ')
CALL SCHOOL
CALL USET ('SYSTEMAXIS')
CALL U3CSYS (70.,70.,0.,0.7,0.7,0.7,0.,0.,0.)
CALL USET ('BLUE')
CALL PIZZA
CALL USET ('SYSTEMAXIS')
CALL U3CSYS (0.,0.,0.,1.,1.,1.,0.,0.,0.)
CALL USET ('BLACK')
CALL ROAD
CALL USET ('SYSTEMAXIS')
RETURN
END
```

```
   SUBROUTINE SCHOOL
   CALL USET ('SOFT')
   CALL USET ('REFERENCEAXIS')
   CALL U3MOVE (0.,0.,0.)
   CALL URECT (50.,25.)
   CALL U3MOVE (3.,4.,0.)
   CALL URECT (7.,6.)
   CALL U3MOVE (13.,4.,0.)
   CALL URECT (17.,6.)
   CALL U3MOVE (47.,4.,0.)
   CALL URECT (43.,6.)
   CALL U3MOVE (37.,4.,0.)
   CALL URECT (33.,6.)
   CALL U3MOVE (25.,0.,0.)
   CALL URECT (22.,6.)
   CALL URECT (28.,6.)
   DO 10 I = 3, 43, 10
   X = I
   CALL U3MOVE (X,10.,0.)
10 CALL URECT (X+4.,20.)
   CALL USET ('WORKINGAXIS')
   CALL U3CSYS (50.,0.,0.,1.,1.,1.,0.,90.,0.)
   ISW = 1
15 CALL U3MOVE (0.,0.,0.)
   CALL URECT (90.,25.)
   DO 20 I = 3, 83, 10
   X = I
   CALL U3MOVE (X,10.,0.)
   CALL URECT (X+4.,20.)
```

```
      IF (I .EQ. 43) GO TO 20
      CALL U3MOVE (X,4.,0.)
      CALL URECT (X+4.,8.)
   20 CONTINUE
      CALL U3MOVE (45.,0.,0.)
      CALL URECT (42.,8.)
      CALL URECT (48.,8.)
      IF (ISW .EQ. 2) GO TO 30
      ISW = 2
      CALL U3CSYS (0.,0.,-90.,1.,1.,1.,0.,-90.,0.)
      GO TO 15
   30 CALL UPSET ('VERTICAL',3.)
      CALL UPSET ('HORIZONTAL',3.)
      CALL UPRINT (0.,21.5,' ROOSEVELT ELEMENTARY SCHOOL;')
      CALL USET ('REFERENCEAXIS')
      CALL UPSET ('ZVALUE',-90.)
      CALL U3MOVE (0.,0.,-90.)
      CALL URECT (60.,25.)
      DO 40 I = 3, 43, 10
      X = I
      CALL U3MOVE (X,4.,-90.)
      CALL URECT (X+4.,8.)
      CALL U3MOVE (X,10.,-90.)
   40 CALL URECT (X+4.,20.)
      CALL UPSET ('ZVALUE',0.)
      RETURN
      END
```

```
      SUBROUTINE CHURCH
      CALL USET ('REFERENCEAXIS')
      CALL U3MOVE (0.,0.,0.)
      CALL URECT (30.,40.)
      CALL U3MOVE (15.,0.,0.)
      CALL URECT (10.,10.)
      CALL URECT (20.,10.)
      CALL U3MOVE (14.,3.,0.)
      CALL URECT (14.25,0.)
      CALL U3MOVE (15.75,3.,0.)
      CALL URECT (16.,0.)
      CALL U3MOVE (0.,40.,0.)
      CALL U3PEN (15.,47.5,0.)
      CALL U3PEN (30.,40.,0.)
      CALL USET ('WORKINGAXIS')
      CALL U3CSYS (30.,0.,0.,1.,1.,1.,0.,90.,0.)
      ISW = 1
   10 CALL U3MOVE (0.,0.,0.)
      CALL URECT (60.,40.)
      DO 20 I = 20, 50, 10
      X = I
      CALL U3MOVE (X,10.,0.)
      CALL U3PEN (X,25.,0.)
      CALL U3PEN (X+4.,30.,0.)
      CALL U3PEN (X+8.,25.,0.)
      CALL U3PEN (X+8.,10.,0.)
   20 CALL U3PEN (X,10.,0.)
      IF (ISW .EQ. 2) GO TO 30
```

```
      ISW = 2
      CALL USCSYS (0.,0.,0.,1.,1.,1.,0.,90.,0.)
      GO TO 10
   30 CALL USET ('REFERENCEAXIS')
      CALL USMOVE (0.,40.,-60.)
      CALL USPEN (15.,47.5,-60.)
      CALL USPEN (30.,40.,-60.)
      CALL USMOVE (0.,0.,-60.)
      CALL USPEN (30.,0.,-60.)
      CALL USMOVE (15.,47.5,-60.)
      CALL USPEN (15.,47.5,0.)
      CALL USMOVE (10.,45.,0.)
      CALL USPEN (10.,45.,-10.)
      CALL USPEN (15.,47.5,-10.)
      CALL USPEN (20.,45.,-10.)
      CALL USPEN (20.,45.,0.)
      CALL USPEN (15.,100.,-5.)
      CALL USPEN (10.,45.,0.)
      CALL USMOVE (10.,45.,-10.)
      CALL USPEN (15.,100.,-5.)
      CALL USPEN (20.,45.,-10.)
      CALL USMOVE (15.,100.,-5.)
      CALL USPEN (15.,110.,-5.)
      CALL USMOVE (13.,107.,-5.)
      CALL USPEN (17.,107.,-5.)
      RETURN
      END
```

```
      SUBROUTINE ROAD
      CALL USMOVE (-200.,0.,0.)
      CALL USPEN (-0.,0.,0.)
      CALL USPEN (-0.,200.,0.)
      CALL USMOVE (200.,0.,0.)
      CALL USPEN (0.,0.,0.)
      CALL USPEN (0.,200.,0.)
      CALL USMOVE (-200.,-0.,0.)
      CALL USPEN (-0.,-0.,0.)
      CALL USPEN (-0.,-200.,0.)
      CALL USMOVE (0.,-200.,0.)
      CALL USPEN (0.,-0.,0.)
      CALL USPEN (200.,-0.,0.)
      RETURN
      END
```

```
      SUBROUTINE PIZZA
      CALL USET ('SOFTWARE CHARACTERS')
      CALL USET ('REFERENCEAXIS')
      CALL U3MOVE (0.,0.,0.)
      CALL U3CSYS (0.,0.,0.,1.,1.,1.,90.,0.,0.)
      CALL U3PEN (40.,0.,0.)
      CALL U3PEN (20.,50.,0.)
      CALL U3PEN (0.,0.,0.)
      CALL U3MOVE (20.,0.,0.)
      CALL URECT (17.5,0.)
      CALL URECT (22.5,0.)
      CALL U3MOVE (10.,2.,0.)
      CALL URECT (15.,0.)
      CALL U3MOVE (30.,2.,0.)
      CALL URECT (25.,0.)
      CALL U3MOVE (13.,10.,0.)
      CALL URECT (27.,15.)
      CALL UPSET ('HORIZONTAL',2.)
      CALL UPSET ('VERTICAL',4.)
      CALL USET ('ITALICS')
      CALL UPRINT (15.65,11.2,'PIZZA;')
      CALL USET ('GOTHIC')
      CALL UPSET ('VERTICAL',.4)
      CALL UPSET ('HORIZONTAL',.3)
      CALL UPRINT (20.95,5.,'IN;')
      CALL USET ('WORKINGAXIS')
      CALL U3CSYS (40.,0.,0.,1.,1.,1.,0.,-160.,0.)




      CALL UPRINT (20.65,5.,'OUT;')
      CALL USET ('REFERENCEAXIS')
      CALL U3MOVE (0.,0.,-60.)
      CALL U3PEN (40.,0.,-60.)
      CALL U3PEN (20.,50.,-60.)
      CALL U3PEN (0.,0.,-60.)
      CALL U3MOVE (16.5,0.,-60.)
      CALL UPSET ('ZVALUE',-60.)
      CALL URECT (21.5,0.)
      CALL UPSET ('ZVALUE',0.)
      CALL USET ('REFERENCEAXIS')
      CALL U3MOVE (0.,0.,0.)
      CALL U3PEN (0.,0.,-60.)
      CALL U3MOVE (40.,0.,-60.)
      CALL U3PEN (40.,0.,0.)
      CALL U3MOVE (20.,50.,0.)
      CALL U3PEN (20.,50.,-60.)
      CALL USET ('WORKINGAXIS')
      CALL USET ('YXZ ')
      CALL U3CSYS (0.,0.,0.,1.,1.,1.,-ATAN2 (50.,20.)*57.29577+90.,
     &             90.,0.)
      CALL USET ('XYZ ')
      CALL UPSET ('HORIZONTAL',2.)
      CALL UPSET ('VERTICAL',2.5)
      CALL UPRINT (5.,5.,'GIANT PIZZA 60;')
      CALL USET ('REFERENCEAXIS')
      RETURN
      END
```

CHAPTER X

GRAPHICAL DATA STRUCTURE PROCESSING

A feature which is contained within 3D GCS is the ability to create, save, and later recreate pictures as they are drawn. It permits the user to specify the description of a commonly used graphic image once, store it, and then invoke this description whenever necessary. This facility is the implementation of an internal high-level GCS data structure.

Before the user can define or use a data structure, he must first provide a work file for GCS to use. This work file can be specified by an UPSET ('LIBRARYFILE', file number) call. The file will then be used to maintain the user's currently active library of GCS graphics data structures.

To build a data structure, the user must first call a routine USTRCT (NAME) which initiates construction of a data structure under the specified name. From then on all calls to the GCS routines listed in Table X-1 will be saved as elements of the data structure. Whenever the specification of the object has been completed, a call is made to UTERM (NAME) to terminate the building process. The resulting data structure is then stored on the library file. During the building process, the build mode may be turned off by specifying 'NOBUILD'. Construction may be resumed by then specifying 'BUILD'. 'BUILD' is automatically specified when USTRCT is called and 'NOBUILD' is set when UTERM is called. Normally, the structure being built will be displayed during the construction process. If the user does not wish to view the structure he may specify 'NOEXECUTE'. If this is done, the structure element will be saved but no visible output will appear. The user may return to viewing his construction process by setting 'EXECUTE' which is the default mode.

To invoke an already created structure, the user uses the U3CALL or UCALL routines specifying the name, position, and orientation of the data structure. The data structure specified will be displayed as indicated. If another structure is currently being built when this call takes place, a U3CALL structure element for the invoked structure will be inserted in the active structure.

When the user terminates the GCS program, he may wish to save his library file so it can be restored at a later time. A routine called UTILTY is used to perform several utility functions concerned with maintaining the library file. To save the current library file, the user simply invokes UTILTY ('SAVE', file number). This file number must be different from the library file. All data structure contained in the library will be reformatted into a standard Hollerith card format and written to the save file. To restore a saved file to the library file, the UTILTY ('LOAD', file number) function would be used. Other UTILTY functions exist to MERGE a save file into the existing library file, PURGE the existing library file, and DELETE or RENAME structures in the library file.

TABLE X-1

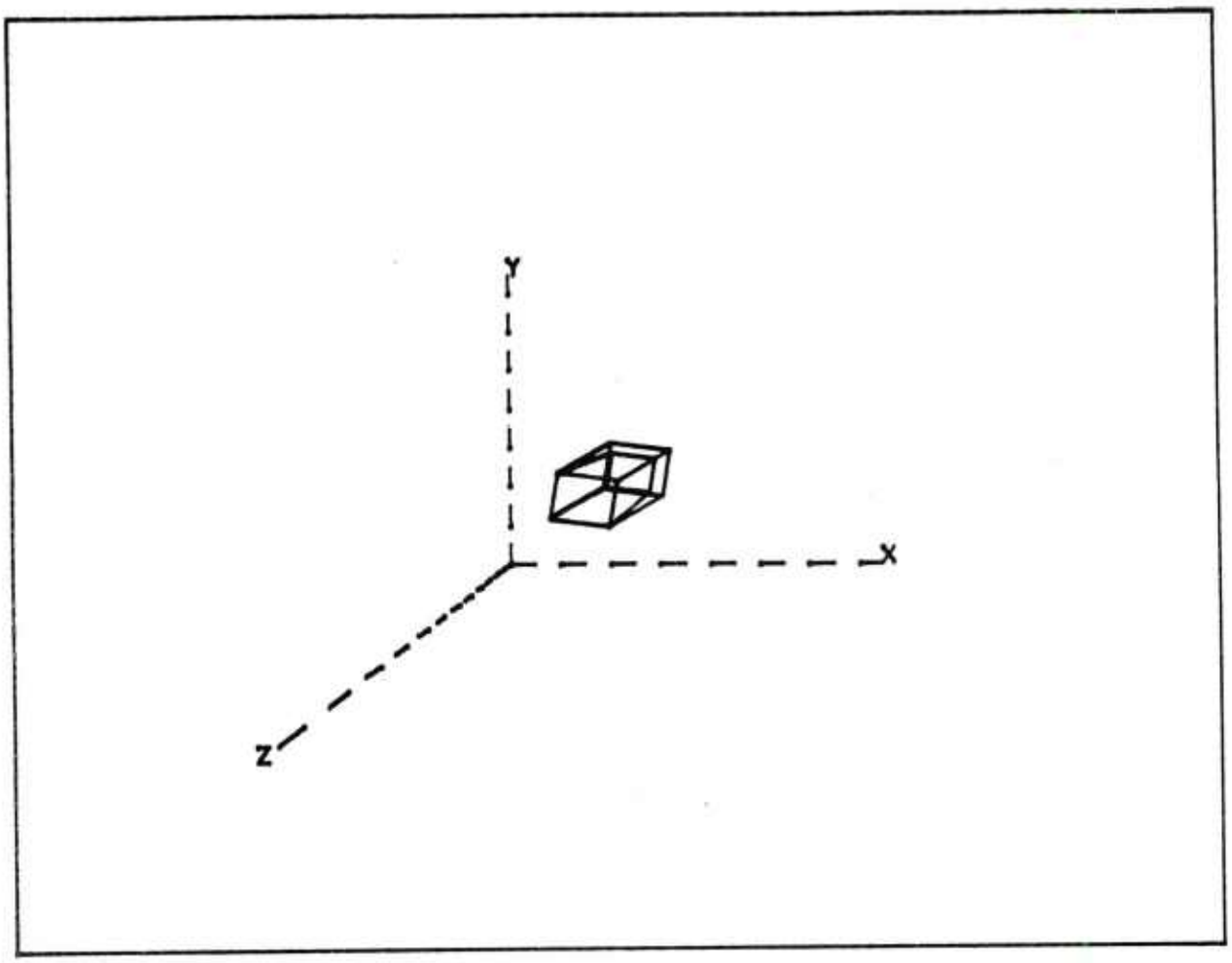
STRUCTURE ELEMENT TABLE

Function	Opcode	No. Real	No. Alpha	Argument Sequence
UARC	1	3	0	R,R,R
U3CALL	2	9	2	R,R,R,R,R,R,R,R,R,A
U3CSYS	3	9	0	R,R,R,R,R,R,R,R,R
UCRCLE	4	3	0	R,R,R
U3DRAW	5	3	0	R,R,R
U3MOVE	6	3	0	R,R,R
U3PEN	7	3	0	R,R,R
U3PRNT	8	4	0	R,R,R,R
UPRNT1	9	1	1	A,R
UPSET	10	1	1	A,R
U3ROTA	11	3	0	R,R,R
U3SCAL	12	3	0	R,R,R
USET	13	0	1	A
U3WRIT	14	4	0	R,R,R,R
UWRIT1	15	1	1	A,R
UPSET	16	0	2	A,A
U3PRNT	17	3	1	R,R,R,A
U3WRIT	18	3	1	R,R,R,A
UPRNT1	19	0	2	A,A
UWRIT1	20	0	2	A,A
URECT	21	2	0	R,R
UFONT	22	0	1	A

```
C       THIS PROGRAM DEMONSTRATES THE SAVING OF DATA STRUCTURES AND
C       SHOWS THE DIFFERENCE BETWEEN ORTHOGONAL AND PERSPECTIVE
C       PLOTTING.
C
C       ATTACH A PERMANENT FILE TO SAVE THE DATA STRUCTURES ON A
C       HONEYWELL COMPUTER.
C
        CALL ATTACH (8,'/TEK3DEMO/SAVE;', 3,0,IST,)
        CALL USTART
        CALL UPSET ('LIBR',1.)
        CALL UPSET ('SETD',1.)
        CALL UPSET ('TERMINATOR', ';')
        CALL USET ('PERC')
        CALL UDAREA (0.,100.,0.,100.)
        CALL UOUTLN
        CALL UWINDO (-100.,100.,-100.,100.)
        CALL USTRCT ('AXIS')
        CALL AXIS
        CALL UTERM ('AXIS')
        CALL USET ('ORTH')
        CALL U3CSYS (25.,25.,25.,1.,1.,1.,10.,10.,0.)
        CALL UVIEW (-20.,-20.,-150.,0.,0.,0.)
        CALL USTRCT ('BOX')
        CALL BOX
        CALL UTERM ('BOX')
        CALL USET ('PERSPECTIVE')
        CALL BOX
        CALL UTILITY ('SAVE',8.)
        CALL UEND
        STOP
        END
        SUBROUTINE AXIS
        CALL USET ('DASH')
C
C       NEED TO DO A MOVE INSIDE STRUCTURE
C
        CALL U3MOVE (-20.,-20.,-20.)
        CALL U3PEN (50.,-20.,-20.)
        CALL UPRNT1 ('X;','TEXT')
        CALL U3MOVE (-20.,-20.,-20.)
        CALL U3PEN (-20.,50.,-20.)
        CALL UPRNT1 ('Y;', 'TEXT')
        CALL U3MOVE (-20.,-20.,-20.)
        CALL U3PEN (-20.,-20.,100.)
        CALL (UPRNT1 ('Z;', 'TEXT')
        RETURN
        END
        SUBROUTINE BOX
        CALL USET ('LINE')
        CALL UMOVE (0.,0.,0.)
        CALL U3PEN (10.,0.,0.)
        CALL U3PEN (10.,10.,0.)
        CALL U3PEN (0.,10.,0.)
        CALL U3PEN (0.,0.,0.)
        CALL U3MOVE (10.,0.,0.)
        CALL U3PEN (10.,0.,60.)
        CALL U3PEN (10.,10.,60.)
        CALL U3PEN (10.,10.,0.)
```

```
CALL U3MOVE (0.,10.,0.)
CALL U3PEN (0.,10.,60.)
CALL U3PEN (10.,10.,60.)
CALL U3MOVE (0.,0.,0.)
CALL U3PEN (0.,0.,60.)
CALL U3PEN (0.,10.,60.)
CALL U3PEN (0.,10.,0.)
CALL U3MOVE (0.,0.,60.)
CALL U3PEN (10.,0.,60.)
RETURN
END
```

EXAMPLE X-1



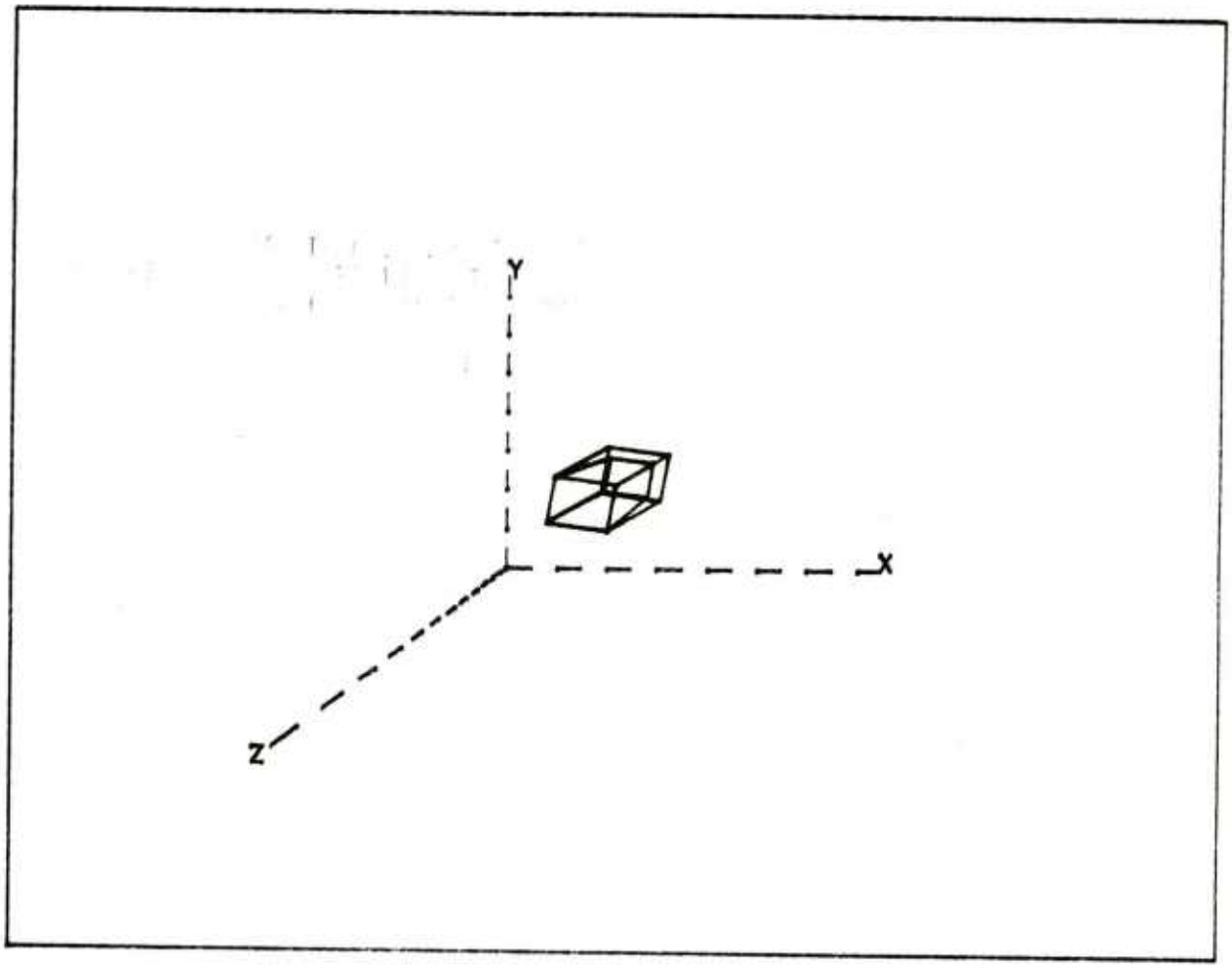

```
CALL ATTACH (6,'/TEK3DEMO/SAVE;',3,0,IST,)
CALL USTART
CALL UPSET ('TERMINATOR',';')
CALL UPSET ('SPEED',120.)
CALL UPSET ('LIBRARY',1.)
CALL UPSET ('SETDASH',1.)
CALL USET ('PERCENTUNITS')
CALL UDAREA (0.,100.,0.,100.)
CALL UERASE
CALL UOUTLN
CALL UWINDO (-100.,100.,-100.,100.)
CALL USTRCT ('AXIS ')
CALL AXIS
CALL UTERM ('AXIS ')
CALL USET ('ORTHOGONAL')
CALL USCSYS (25.,25.,25.,1.,1.,1.,10.,10.,0.)
CALL UVIEW (-20.,-20.,-150.,0.,0.,0.)
CALL USTRCT ('BOX  ')
CALL BOX
CALL UTERM ('BOX  ')
CALL USET ('PERSPECTIVE')
CALL BOX
CALL UTILTY ('SAVE',0.)
CALL UEND
STOP
END
```

```
SUBROUTINE AXIS
CALL USET ('DASH')
CALL U3MOVE (-20.,-20.,-20.)
CALL U3PEN (50.,-20.,-20.)
CALL UPRNT1 ('X','TEXT')
CALL U3MOVE (-20.,-20.,-20.)
CALL U3PEN (-20.,50.,-20.)
CALL UPRNT1 ('Y','TEXT')
CALL U3MOVE (-20.,-20.,-20.)
CALL U3PEN (-20.,-20.,100.)
CALL UPRNT1 ('Z','TEXT')
RETURN
END
```

```
SUBROUTINE BOX
CALL USET ('LINE')
CALL U3MOVE (0.,0.,0.)
CALL U3PEN (10.,0.,0.)
CALL U3PEN (10.,10.,0.)
CALL U3PEN (0.,10.,0.)
CALL U3PEN (0.,0.,0.)
CALL U3MOVE (10.,0.,0.)
CALL U3PEN (10.,0.,60.)
CALL U3PEN (10.,10.,60.)
CALL U3PEN (10.,10.,0.)
CALL U3MOVE (0.,10.,0.)
CALL U3PEN (0.,10.,60.)
CALL U3PEN (10.,10.,60.)
CALL U3MOVE (0.,0.,0.)
CALL U3PEN (0.,0.,60.)
CALL U3PEN (0.,10.,60.)
CALL U3PEN (0.,10.,0.)
CALL U3MOVE (0.,0.,60.)
CALL U3PEN (10.,0.,60.)
RETURN
END
```

```
C     THIS PROGRAM DEMONSTRATES THE LOADING OF DATA STRUCTURES
C     THAT WERE PREVIOUSLY SAVED ON A PERMANENT FILE ON A HONEYWELL
C     COMPUTER.
C
      CALL ATTACH (8,'/TEK3DEMO/SAVE;',3,0,IST,)
      CALL USTART
      CALL UPSET ('LIBR',1.)
      CALL UPSET ('SETD',1.)
      CALL USET ('PERC')
      CALL UDAREA (0.,100.,0.,100.)
      CALL UOUTLN
      CALL UWINDO (-100.,100.,-100.,100.)
      CALL UTILTY ('LOAD',8.)
      CALL U3MOVE (0.,0.,0.)
      CALL UINVOK ('AXIS')
      CALL USET ('ORTH')
      CALL U3CSYS (25.,25.,25.,1.,1.,1.,10.,10.,0.)
      CALL UVIEW (-20.,-20.-150.,0.,0.,0.)
      CALL U3MOVE (0.,0.,0.)
      CALL UINVOK ('BOX')
      CALL USET ('PERSPECTIVE')
      CALL U3MOVE (0.,0.,0.)
      CALL UNIVOK ('BOX')
      CALL UEND
      STOP
      END
```

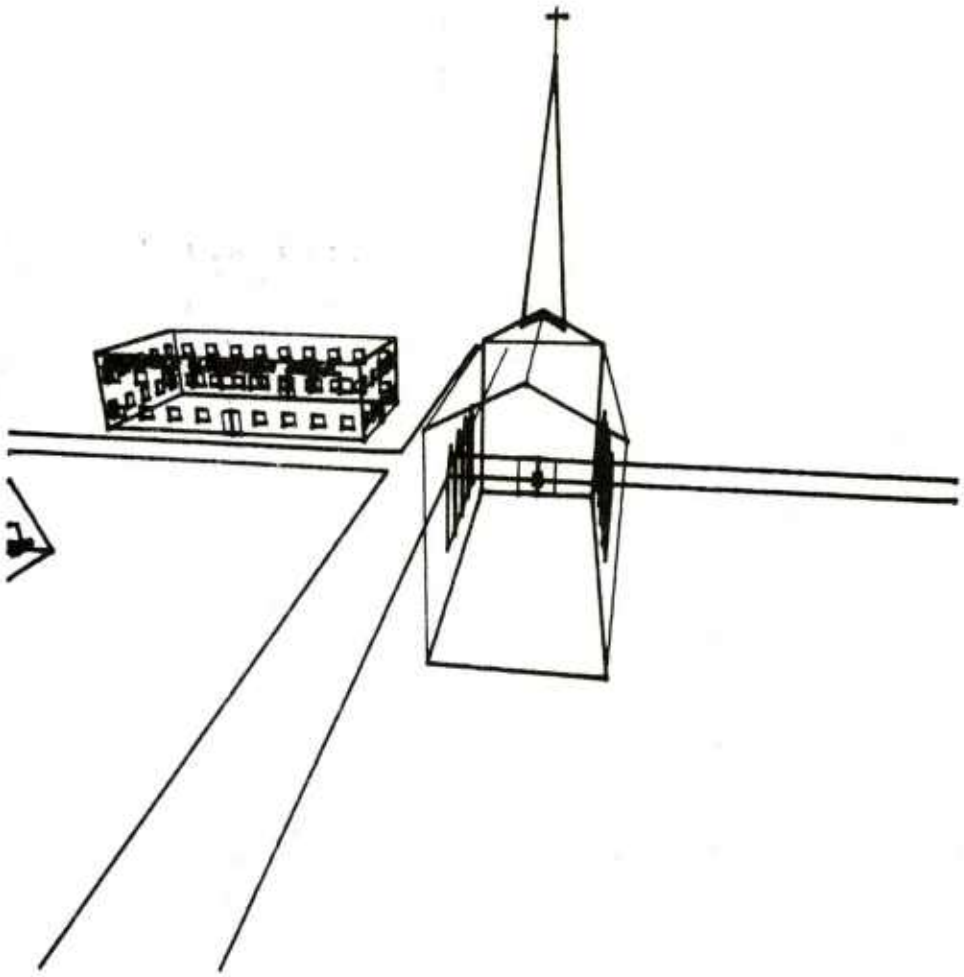
EXAMPLE X-2




```
      CALL ATTACH (8,'/TEK3DEMO/SAVE,',3,0,IST,)
      CALL USTART
      CALL UPSET ('LIBRARY',1.)
      CALL UPSET ('TERMINATOR',',')
      CALL UPSET ('SPEED',120.)
      CALL USET ('PERCENTUNITS')
      CALL UDAREA (0.,100.,0.,100.)
      CALL UERASE
      CALL UOUTLN
      CALL UWINDO (-100.,100.,-100.,100.)
      CALL UTILTY ('LOAD',6.)
      CALL U3MOVE (0.,0.,0.)
      CALL UINVOK ('AXIS ')
      CALL USET ('ORTHOGONAL')
      CALL U3CSYS (25.,25.,25.,1.,1.,1.,10.,10.,0.)
      CALL UVIEW (-20.,-20.,-150.,0.,0.,0.)
      CALL U3MOVE (0.,0.,0.)
      CALL UINVOK ('BOX  ')
      CALL USET ('PERSPECTIVE')
      CALL U3MOVE (0.,0.,0.)
      CALL UINVOK ('BOX  ')
      CALL UEND
      STOP
      END
```

```
C     THIS PROGRAM DEMONSTRATES THE LOADING OF DATA STRUCTURES
C     THAT WERE PREVIOUSLY SAVED ON A PERMANENT FILE ON A HONEYWELL
C     COMPUTER FOR DISPLAY ON TEKTRONIX 4014/4015 TERMINAL.
C
      CALL ATTACH (8,'/TEK3DEMO/SAVE;',3,0,IST,)
      CALL USTART
      CALL UPSET ('LIBR',1.)
      CALL UTILITY ('LOAD',8.)
      CALL UPSET ('TERM','<')
      CALL USET ('VIEW')
      CALL UVWPRT (150.)
      CALL UWINDO (-100.,100.,-100.,100.)
      CALL UVIEW (-40.,200.,70.,-20.,20.,0.)
      CALL UDAREA (0.,7.,0.,7.)
      CALL VILLAG
      CALL UDAREA (7.1,14,1.0.,7.)
      CALL VIEW (-70.,-160.,50.,0.,0.,10.)
      CALL VILLAG
      CALL UEND
      STOP
      END
      SUBROUTINE VILLAG
      CALL USET ('XYZ')
      CALL USET ('SYST')
      CALL USET ('REFE')
      CALL USET ('BLACK')
      CALL U3CALL (-50.,20.,0.,1.,1.,1.,90.,0.,0.,'CHURCH')
      CALL USET ('RED')
      CALL U3CALL (24.,-19.,0.,1.,1.,1.,90.,-90.,0.,'SCHOOL')
      CALL USET ('BLUE')
      CALL U3CALL (70.,70.,0.,0.7,0.7,0-.7,0.,0.,0.,'PIZZA')
      CALL USET ('BLACK')
      CALL U3CALL (0.,0.,0.,1.,1.,1.,0.,0.,0.,'ROAD')
      RETURN
      END
```

EXAMPLE X-3



```
CALL ATTACH (8,'/TEK3DEMO/SAVE,',9,0,IST,)
CALL USTART
CALL UPSET ('LIBRARY',1.)
CALL UTILTY ('LOAD',8.)
CALL UPSET ('TERMINATOR',',')
CALL USET ('VIEWDISTANCE')
CALL UVWPRT (150.)
CALL UWINDO (-100.,100.,-100.,100.)
CALL UVIEW (-40.,200.,70.,-20.,20.,0.)
CALL VILLAG
CALL UEND
STOP
END
SUBROUTINE VILLAG
CALL USET ('XYZ ')
CALL USET ('SYSTEMAXIS')
CALL USET ('REFERENCEAXIS')
CALL USET ('BLACK')
CALL USCALL (-50.,20.,0.,1.,1.,1.,90.,0.,0.,'CHURCH')
CALL USET ('RED ')
CALL USCALL (24.,-10.,0.,1.,1.,1.,90.,-90.,0.,'SCHOOL')
CALL USET ('BLUE')
CALL USCALL (70.,70.,0.,0.7,0.7,0.7,0.,0.,0.,'PIZZA')
CALL USET ('BLACK')
CALL USCALL (0.,0.,0.,1.,1.,1.,0.,0.,0.,'ROAD ')
RETURN
END
```

CHAPTER XI

PICTURE SEGMENTATION AND NAMING

Refresh Graphic Facilities

Many computer graphics display applications require only a static display in order to convey information. That is, the complete picture provides all of the relevant information that can be derived from the data used to produce the display. In some situations however, a static display provides incomplete information. For example, the plot of a missile trajectory does not reveal the relative speed of the projectile while in flight. In such situations, a dynamic display which portrays motion provides more information to the user than does a static display. GCS provides a facility for the dynamic display of information and for selective erase of information on refresh graphic display devices if available. The facility allows the user to identify and bracket a portion or portions of his display, assign a name to the section, turn the named section on or off, and replace a named section with a new definition of the section.

The basic unit of information of a dynamic display is the 'frame'. A frame represents a portion of a display having a known origin and a known termination. The GCS subroutines which delimit the start and the end of a frame are UFRAME and UFREND. They are invoked as follows:

```
CALL UFRAME (NAME)
CALL UFREND (NAME)
```

where NAME is an eight character alphanumeric variable or constant. The subroutine UFRAME indicates the starting point of the framed information and subroutine UFREND denotes the ending point. Note that multiple frames may be defined, but nesting of frames is not permitted; that is, subroutine UFREND must be called to 'close' the current frame before the subroutine UFRAME is invoked to 'open' another frame.

Whenever a UFRAME/UFREND pair are invoked to replace the previous occurrence of the named frame, the previous occurrence is deleted upon completion of the new version of the frame, as signalled by the call to UFREND for that frame. Thus the complete frame will replace the previous complete frame. This is the default condition in GCS and is known as 'invisible' building of a frame. It can be specified by the following invocation to USET:

```
CALL USET ('INVISIBLE')
```

For some dynamic graphic applications, it is preferable to see the frame being built line-by-line. In this case, the previous frame must be deleted when the call to UFRAME is made for frame redefinition. The user can request the visible building of a frame by the following call:

```
CALL USET ('VISIBLE')
```

After a frame is completely built, and terminated by a call to UFREND, it may be selectively 'turned on' and 'turned off' by the use of the following GCS subroutines:

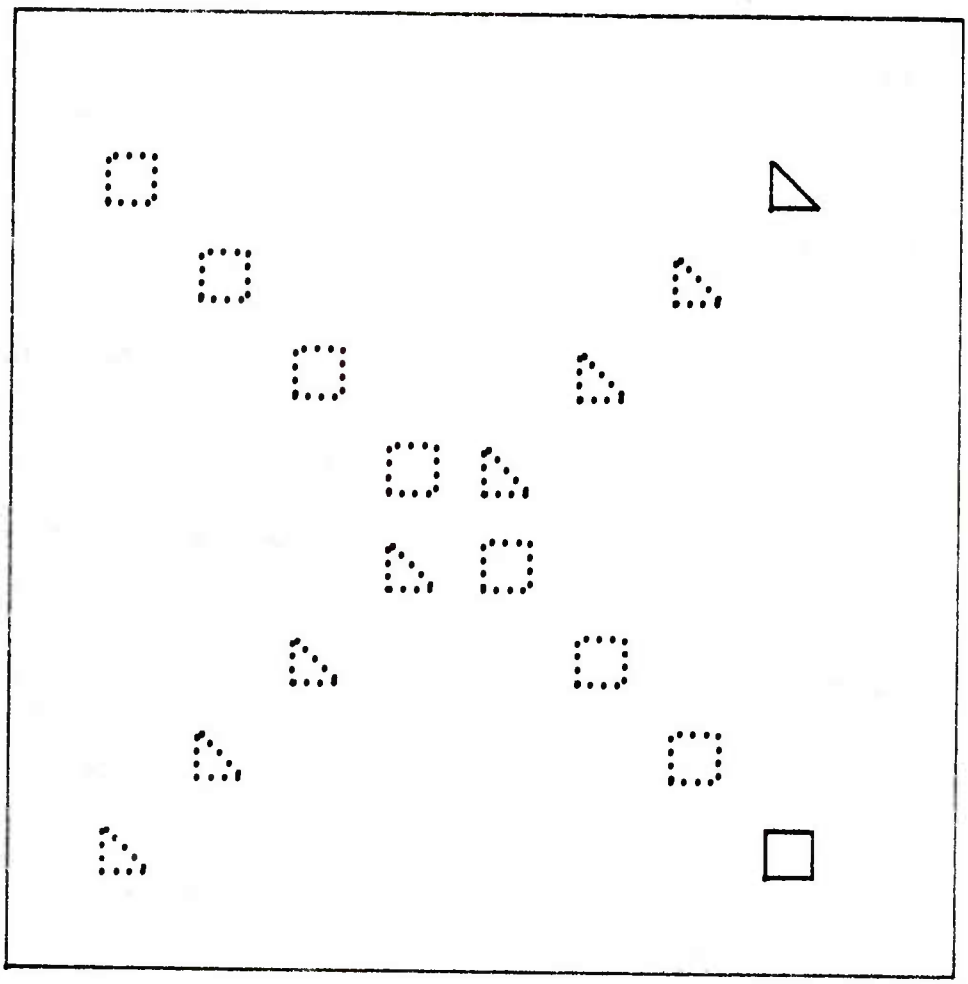
```
CALL USHOW (NAME)
CALL UNSHOW (NAME)
```

where NAME is the eight character alphanumeric variable or constant which is the name assigned to the frame upon which the operation is to be performed. When USHOW is invoked for an invisible frame, the frame is made visible; if UNSHOW is invoked for a visible frame, the frame is made invisible.

Note that the GCS subroutine UERASE affects the status of frames by completely removing all frames and frame names from GCS. Thus if USHOW or UNSHOW are invoked after a call to UERASE, an error will be indicated. In addition, it is improper to call UERASE while a frame is 'open', i.e., after a call to UFRAME and before a call to UFREND.

```
C     SAMPLE PROGRAM USED TO ILLUSTRATE MULTIPLE ENTITY FRAMING.
C
C     INITIALIZE VARIABLES, ENTER GCS, OUTLINE VIRTUAL WINDOW.
C
      X=10.0
      Y=80.0
      CALL USTART
      CALL UOUTLN
      CALL UPSET ('LIBRARY', 1.)
C
C     DEFINE A LOOP TO DRAW A TRIANGLE AND SQUARE DYNAMICALLY.
C
      DO 1 I=1,8
C
C     INDICATE THE BEGINNING OF THE TRIANGLE FRAME DEFINITION, THEN
C     PROVIDE TEN COMMANDS REQUIRED TO DRAW THE FIGURE.
C
      CALL UFRAME ('TRIANGLE')
      CALL UMOVE (X,X)
      CALL UPEN (X,(X+5.0))
      CALL UPEN ((X+5.0),X)
      CALL UPEN (X,X)
C
C     INDICATE THE TERMINATION OF THE TRIANGLE FRAME, AND THE
C     BEGINNING OF THE FRAME DEFINITION FOR THE SQUARE.
C
      CALL UFREND ('TRIANGLE')
      CALL UFRAME ('SQUARE')
C
C     NOW PROVIDE THE PEN COMMANDS REQUIRED TO DRAW A SQUARE.
C
      CALL UMOVE (X,Y)
      X=X+10.0
      Y=Y-10.0
      CALL URECT ((X-5.0),Y+15.0))
C
C     INDICATE THE TERMINATION OF THE SQUARE FRAME, AND LOOP.
C
      CALL UFREND ('SQUARE')
    1 CONTINUE
      CALL UEND
      STOP
      END
```

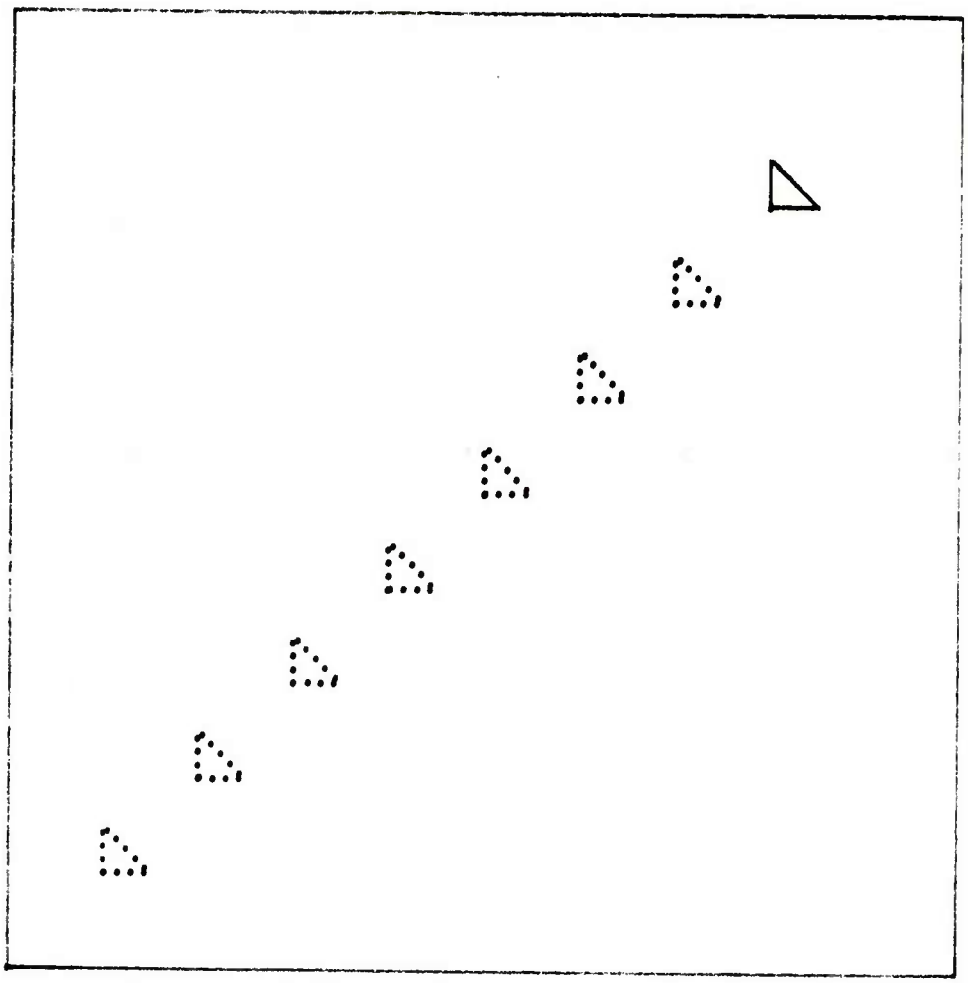
EXAMPLE XI-1



```
      X = 10.0
      Y = 80.0
      CALL USTART
      CALL UPSET ('LIBRARY',1.)
      CALL UOUTLN
      DO 1 I = 1, 8
      CALL UFRAME ('TRIANGLE')
      CALL UMOVE (X,X)
      CALL UPEN (X,(X+5.0))
      CALL UPEN ((X+5.0),X)
      CALL UPEN (X,X)
      CALL UFREND ('TRIANGLE')
      CALL UFRAME ('SQUARE')
      CALL UMOVE (X,Y)
      X = X + 10.0
      Y = Y - 10.0
      CALL URECT ((X-5.0),(Y+15.0))
      CALL UFREND ('SQUARE')
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE GCS FRAMING FACILITIES.
C
C       DEFINE INITIAL POSITION, ENTER GCS, OUTLINE VIRTUAL WINDOW.
C
        CHARACTER CHAR*1
        X=10.0
        CALL USTART
        CALL UOUTLN
        CALL UPSET ('LIBRARY',1.)
C
C       DEFINE A LOOP TO DYNAMICALLY DISPLAY 8 DEFINITIONS OF A TRIANGLE.
C       INDICATE THE START OF THE 'FRAMED' INFORMATION.
C
        DO 1 I=1,8
        CALL UFRAME ('TRIANGLE')
C
C       PEN COMMANDS TO DEFINE THE TRIANGLE.  A POSITIONAL VARIABLE IS
C       ALSO UPDATED WITHIN THIS LOOP.
C
        CALL UMOVE (X,X)
        CALL UPEN (X,(X+5.0))
        CALL UPEN ((X+5.0),X)
        CALL UPEN (X=X+10.0
C
C       INDICATE THE TERMINATION OF THE TRIANGLE FRAME DEFINITION.
C
        CALL UFREND ('TRIANGLE')
    1   CONTINUE
C       'DEACTIVATE' THE EIGHTH FRAME DEFINITION OF THE TRIANGLE THAT WAS
C       DEFINED; I.E., MAKE IT 'INVISIBLE'.
C
        CALL UNSHOW ('TRIANGLE')
C
C       REQUEST A CHARACTER FROM THE KEYBOARD; ACTIVATE THE LAST
C       FRAME IF THE LETTER 'T' IS ENTERED; OTHERWISE, TERMINATE.
C
    2   CALL UAIN (CHAR)
        IF (CHAR.NE.'T') GO TO 3
        CALL USHOW ('TRIANGLE')
        GO TO 2
    3   CALL UEND
        STOP
        END
```

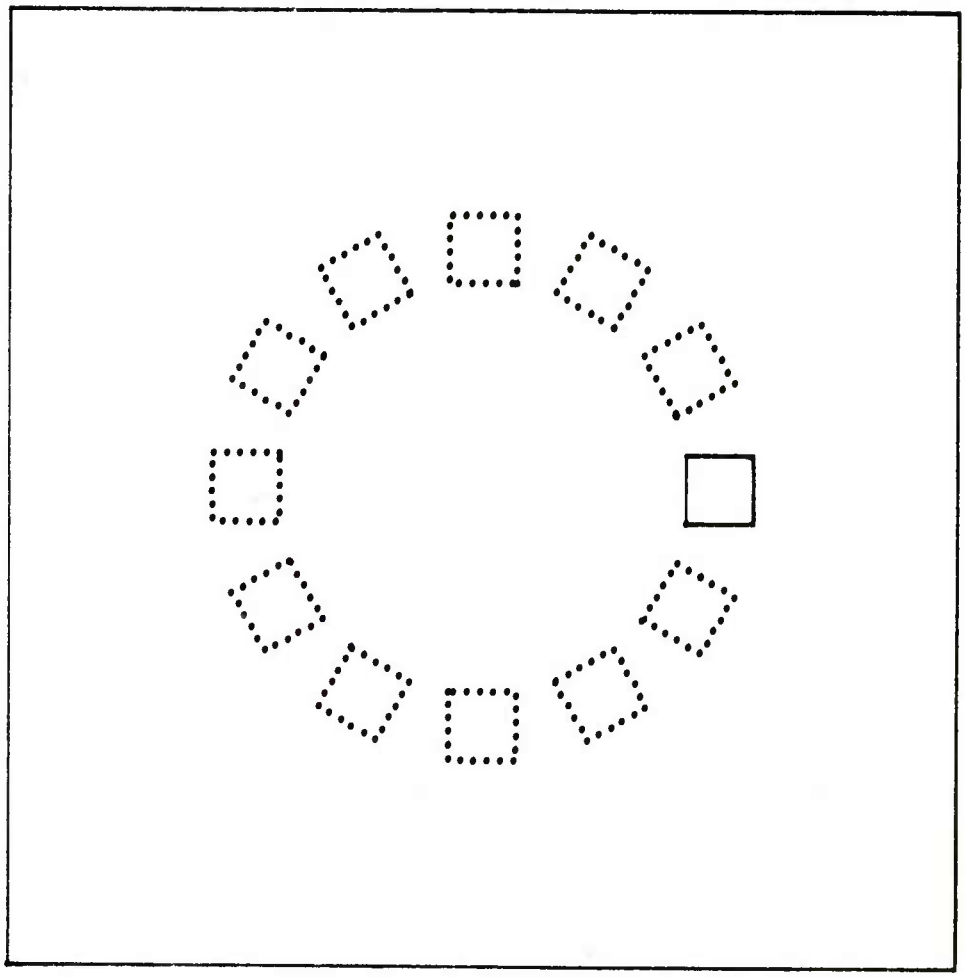
EXAMPLE XI-2



```
      CHARACTER CHAR*1
      X=10.0
      CALL USTART
      CALL UPSET ('LIBRARY',1.)
      CALL UOUTLN
      DO 1 I = 1, 6
      CALL UFRAME ('TRIANGLE')
      CALL UMOVE (X,X)
      CALL UPEN (X,(X+5.0))
      CALL UPEN ((X+5.0),X)
      CALL UPEN (X,X)
      X = X + 10.0
      CALL UFREND ('TRIANGLE')
    1 CONTINUE
      CALL UNSHOW ('TRIANGLE')
    2 CALL UAIN (CHAR)
      IF (CHAR .NE. 'T') GO TO 3
      CALL USHOW ('TRIANGLE')
      GO TO 2
    3 CALL UEND
      STOP
      END
```

```
C       SAMPLE PROGRAM USED TO ILLUSTRATE ELEMENTARY FRAMING AND
C       ROTATION.  THE ROTATIONAL TECHNIQUE EMPLOYED IN THE PROGRAM IS
C       QUITE LIMITED, HENCE THE USER IS DIRECTED TO USE THE GCS
C       SUBROUTINE 'UCOSYS' FOR APPLICATIONS REQUIRING MORE ROTATIONAL
C       CAPABILITIES.
C
C       INITIALIZE VARIABLES, ENTER GCS, DEFINE NEW VIRTUAL WINDOW, AND
C       OUTLINE IT.
C
        DATA DEGREE/0.0/
        CALL USTART
        CALL UWINDO (-50.,50.,-50.,50.)
        CALL UOUTLN
        CALL UPSET ('LIBRARY', 1.)
C
C       SPECIFY THAT POLAR COORDINATES ARE TO BE USED TO SIMPLIFY THE
C       PROGRAM, AND DEFINE LOOP TO DRAW A SQUARE DYNAMICALLY.
C
        CALL USET ('POLAR')
        DO 1 I=1,12
        DEGREE=DEGREE=30.0
C
C       INDICATE THE BEGINNING OF FRAME DEFINITION FOR THE SQUARE, AND
C       PROVIDE THE NECESSARY COMMANDS TO ROTATE AND DRAW IT.
C
        CALL UFRAME ('SQUARE')
        CALL UMOVE (25.,DEGREE)
        CALL UPSET ('ROTATE',DEGREE)
        CALL USET ('RELATIVE')
        CALL UPLYGN (0.,0.,4.,5.)
        CALL USET ('ABSOLUTE)
C
C       INDICATE THE TERMINATION OF THE SQUARE FRAME, AND THE LOOP.
C
        CALL UFREND ('SQUARE')
    1   CONTINUE
        CALL UEND
        STOP
        END
```

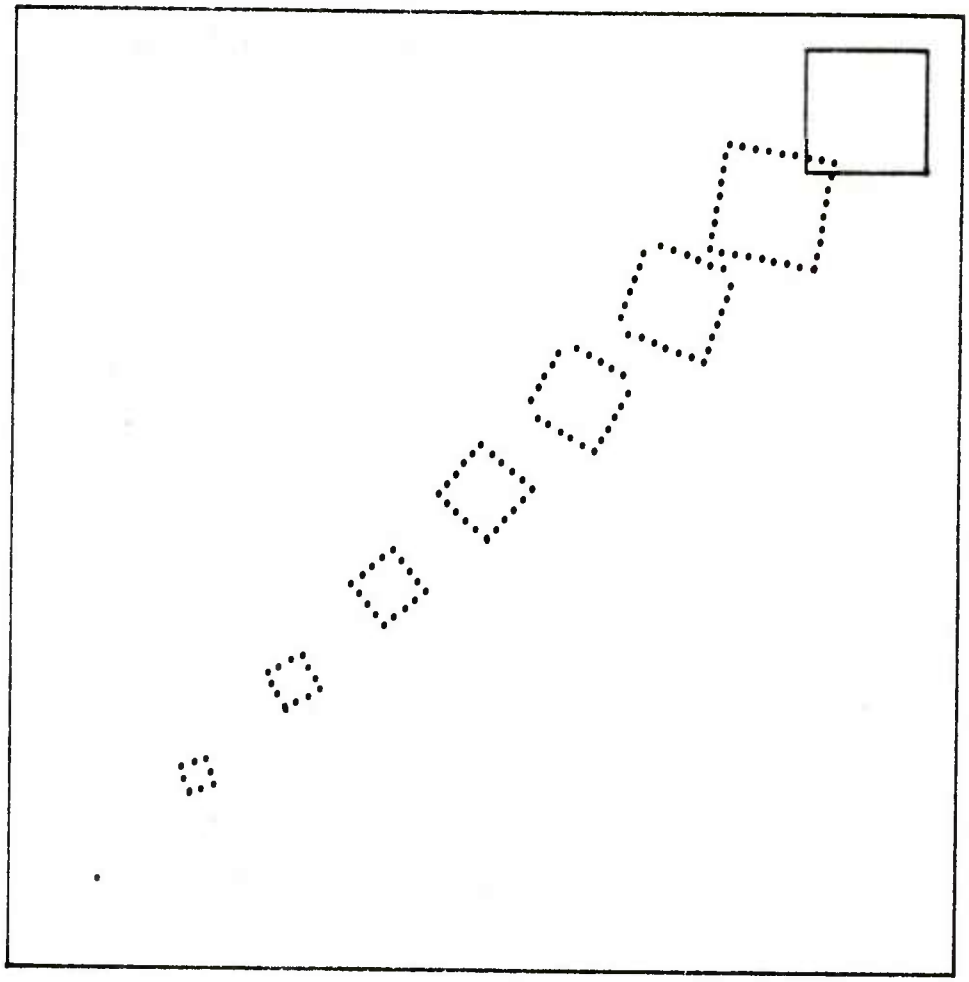
EXAMPLE XI-3



```
      DATA DEGREE/0.0/
      CALL USTART
      CALL UPSET ('LIBRARY',1.)
      CALL UWINDO (-50.,50.,-50.,50.)
      CALL UOUTLN
      CALL USET ('POLAR')
      DO 1 I = 1, 12
      DEGREE = DEGREE + 30.0
      CALL UFRAME ('SQUARE')
      CALL UMOVE (25.,DEGREE)
      CALL UPSET ('ROTATE',DEGREE)
      CALL USET ('RELATIVE')
      CALL UPLYGN (0.,0.,4.,5.)
      CALL USET ('ABSOLUTE')
      CALL UFREND ('SQUARE')
    1 CONTINUE
      CALL UEND
      STOP
      END
```

```
C     SAMPLE PROGRAM ILLUSTRATING APPLICATIONS OF GCS SUBROUTINES
C     'USHOW' AND 'UNSHOW'.  A SQUARE WILL BE SIMULTANEOUSLY SCALED,
C     ROTATED (ORIENTED), AND TRANSLATED IN NINE DISCRETE PHASES, WITH
C     A SEPARATE FRAME REPRESENTING EACH PHASE.
C
C     INITIALIZE VARIABLES, ENTER GCS, AND OUTLINE VIRTUAL WINDOW.
C
      CHARACTER FRAME*8(9)
      DATA FRAME/'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I'/
      CALL USTART
      CALL UOUTLN
      CALL UPSET ('LIBRARY',1.)
C
C     SETUP LOOP TO DEFINE EACH OF THE NINE FRAMES FOR THE SQUARE.
C
      DO 1 I=1,9
      X=10.0*FLOAT(I)
C
C     INDICATE THE BEGINNING OF FRAME DEFINITION FOR THE SQUARE, AND
C     PROVIDE COMMANDS TO SCALE, ROTATE, AND TRANSLATE IT.
C
      CALL UFRAME (FRAME(I))
      CALL UMOVE X,X)
      CALL UPSET ('ROTATE', X)
      CALL USET ('RELATIVE')
      CALL UPLYGN (0.,0.,4.,FLOAT(I))
      CALL USET ('ABSOLUTE')
C
C     INDICATE THE TERMINATION OF EACH FRAME, AND CALL UNSHOW TO MAKE
C     THE NEWLY DEFINED FRAME INVISIBLE.
C
      CALL UFREND (FRAME(I))
      CALL UNSHOW (FRAME (I))
    1 CONTINUE
C
C     AT THIS POINT, ALL OF THE FRAMES HAVE BEEN DEFINED, NOW WE WILL
C     DISPLAY EACH OF THE FRAMES IN SUCCESSION, FOR A TOTAL NUMBER OF
C     90 CYCLES = 10 OCCURRENCES OF EACH FRAME.
C
      DO 2 I=1,90
      J=MOD((I-1),9)+1
C
C     CALL UNSHOW TO DISPLAY THE 'J'TH FRAME (MAKE IT VISIBLE), AND THEN
C     CALL UNSHOW TO MAKE IT INVISIBLE. SINCE THIS IS DONE AT RAPID RATE,
C     THE SQUARE WILL APPEAR TO BE DYNAMICALLY SCALED, ROTATED, AND
C     TRANSLATED.
C
      CALL UNSHOW (FRAME(J))
      CALL UNSHOW (FRAME(J))
    2 CONTINUE
C     INDICATE END OF ALL GRAPHIC ACTIVITY AND TERMINATE PROGRAM.
C
      CALL UEND
      STOP
      END
```

EXAMPLE XI-4



```
      CHARACTER FRAME*8(9)
      DATA FRAME/'A','B','C','D','E','F','G','H','I'/
      CALL USTART
      CALL UPSET ('LIBRARY',1.)
      CALL UOUTLN
      DO 1 I = 1, 9
      X = 10.0 * FLOAT(I)
      CALL UFRAME (FRAME(I))
      CALL UMOVE (X,X)
      CALL UPSET ('ROTATE',X)
      CALL USET ('RELATIVE')
      CALL UPLYGN (8.,0.,4.,FLOAT(I))
      CALL USET ('ABSOLUTE')
      CALL UFREND (FRAME(I))
      CALL UNSHOW (FRAME(I))
1     CONTINUE
      DO 2 I = 1, 99
      J = MOD((I-1),9) + 1
      CALL USHOW (FRAME(J))
      CALL UNSHOW (FRAME(J))
2     CONTINUE
      CALL UEND
      STOP
      END
```

APPENDIX A

ALPHABETICAL LISTING OF GCS SUBROUTINES

UAIN	Accepts one character from the terminal
	CALL UAIN (ICHAR)
UALPHA	Insures that terminal is in alphanumeric mode
	CALL UALPHA
UAOUT	Outputs a character at current pen position subject to margining
	CALL UAOUT (ICHAR)
UAPEND	Adds GCS string terminator to character string
	CALL UAPEND (COUNT,DATAIN,DATOUT)
UARC	Draws an arc from current pen position
	CALL UARC (X,Y,ANGLE)
UASPCT	Forces the display dimensions to satisfy the specified aspect ratio
	CALL UASPCT (RATIO)
UAVERG	Fits a moving average curve to time series data
	CALL UAVERG (ARRAY,POINTS,FCST,PERIOD)
UAXIS	Draws axes with appropriate numeric and alphanumeric labeling
	CALL UAXIS (XMIN,XMAX,YMIN,YMAX)
UBAR	Draws a bar chart with appropriate numeric and alphameric labels
	CALL UBAR (ARRAY,PTS,LABELS,SIZE)
UBELL	Sounds the audible alarm at the terminal
	CALL UBELL
UCALL	Invokes a graphic data structure in two dimensions
	CALL UCALL (NAME,DX,DY,SX,SY,ANGLE)
UCHART	Draws a grouped bar chart for multi-valued data
	CALL UCHART (ARRAY,GROUPS,BARS,LABELS,YMAXL)
UCLOSE	Closes the current open frame/segment
	CALL UCLOSE (SEGNAM)
UCOLOR	Defines entries in a program modifiable table of colors

CALL UCOLOR (CLRIDX, CLRCNT, CLRNAM, CLRVAL)

UCONIC Draws generalized conic sections

CALL UCONIC (X,Y,P,E,THETA1,THETA2)

UCONTR Draws contours on regular array of data

CALL UCONTR (Z,X,Y,A,FX,FY,CURVE,FN)

UCOSYS Creates a user coordinate plotting system

CALL UCOSYS (DX,DY,SX,SY,ANGLE)

UCOUNT Counts number of characters in character string

CALL UCOUNT (DATA,COUNT)

UCRCLE Draws a circle whose center location and radius are specified

CALL UCRCLE (X,Y,RADIUS)

UDAREA Sets the device display area associated with user window

CALL UDAREA (XMIN,XMAX,YMIN,YMAX)

UDELAL Deletes all currently defined frame/segments

CALL UDELAL

UDELET Deletes a currently-defined frame/segment

CALL UDELET (SEGNAM)

UDIMEN Adjusts physical boundaries of output device (alters aspect ratio)

CALL UDIMEN (XMAX,YMAX)

UDOIT Perform various page layout functions

CALL UDOIT (ACTION)

UDRAW Draws solid line vector

CALL UDRAW (X,Y)

UDRIN Performs the input requested graphic operation and returns request

CALL UDRIN (X,Y,ICHAR)

UEND Terminates graphic operations and positions pen in home position

CALL UEND

UERASE Erases the screen or requests a clean plotting surface

CALL UERASE

UERROR Returns listing of source records with GCS error commentary

CALL UERROR (ERLAST,TOTAL)

UFLUSH Insures that visual display reflects all net program graphical output

CALL UFLUSH

UFONT Changes the text font

CALL UFONT (NAMFNT)

UFORMT Configures the display surface to the requested format

CALL UFORMT (FORMAT)

UFRAME Defines the start of a named set of graphical commands

CALL UFRAME (NAME)

UFREND Defines the end of a named set of graphical commands

CALL UFREND (NAME)

UGRIN Gets coordinates and a character from terminal and returns them

CALL UGRIN (X,Y,ICHAR)

UHDCPY Generate a hardcopy

CALL UHDCPY

UHISTO Draws a histogram with appropriate numeric and alphanumeric labels

CALL UHISTO (ARRAY,PTS,BARS)

UHOME Moves beam to home position

CALL UHOME

UIMAGE Applies general 2-D image transformations to 'retained' frames/segments

CALL UIMAGE (X,Y,SX,SY,R,SEGNAM)

UINPUT Inputs alphanumeric information from the current position

CALL INPUT (DATA,COUNT,FLAG,OPTION)

UINVOK Invokes a GCS structure at the current position

CALL UINVOK (NAME)

ULINE Connects two arrays of points with current line option

CALL ULINE (X,Y,PTS)

ULINFT Determines linear least squares fit to points provided

CALL ULINFT (X,Y,XN,S,YI)

ULOOK — Establish portion display area onto which corresponding portion of current virtual space viewport will be mapped

CALL ULOOK (XMIN,XMAX,YMIN,YMAX)

ULSTSQ — Calculates least squares polynomial fit to points provided

CALL ULSTSQ (X,Y,XN,COEFF)

UMARGN — Sets the left and right, top and bottom alphanumeric window boundaries

CALL UMARGN (XLEFT,XRIGHT,YBOTTM,YTOP)

UMENU — Menu board generating routine

CALL UMENU (POINTS,LABELS,CHOICE)

UMODFY — Modifies setting of frame/segment attributes

CALL UMODFY (SEGNAM,NAMAT,ATVALU)

UMOVE — Moves the pen to position specified by input arguments

CALL UMOVE (X,Y)

UNSAVE — Restores all variables of the graphic status area

CALL UNSAVE (ARRAY)

UNSHOW — Causes the named frame/segment of graphical information to be made invisible

CALL UNSHOW (NAME)

UNSVPN — Restores all pen related variables in the graphics status area

CALL UNSVPN (ARRAY)

UNSVTR — Restores coordinate system related variables in the graphics status area

CALL UNSVTR (ARRAY)

UOPEN — Opens a frame/segment

CALL UOPEN (SEGNAM,SEGTYP)

UORIGN — Creates a user coordinate system at the current beam/pen position

CALL UORIGN

UOUTLN — Draws a box around the user's display area

CALL UOUTLN

UPAUSE — Suspends execution until one character is entered from keyboard

CALL UPAUSE

UPEN	Draws a line from current pen position to given coordinates
	CALL UPEN (X,Y)
UPEN1	Sets one 'USET' option for this call only before executing pen movement
	CALL UPEN1 (X,Y,OPTION)
UPIE	Draws a pie chart with appropriate numeric and alphameric labels
	CALL UPIE (ARRAY,PTS,LABELS,SIZE)
UPLACE	Applies 2-D translation image transformation to 'retained' frames/segments
	CALL UPLACE (X,Y,SEGNAM)
UPLOT	General purpose multi-curve plotting routine
	CALL UPLOT (X,Y,CURVES,PARRAY,OPTION)
UPLOT1	Plots a single curve
	CALL UPLOT1 (X,Y,PTS)
UPLYGN	Draws a regular polygon
	CALL UPLYGN (X,Y,PTSIN,RADIUS)
UPOINT	Defines point which, together with two end points of a given line, defines the plane for the terminator and tic line
	CALL UPOINT (X,Y,Z)
UPOST	Insures that only defined, visible frame/segments are displayed
	CALL UPOST
UPRINT	Prints information in hardware or software characters
	CALL UPRINT (X,Y,INPUT)
UPRNT1	Allows alphanumeric output at current position with specified option
	CALL UPRNT1 (DATA,OPTION)
UPSET	Changes setting in the GSA which requires a parameter value to be set
	CALL UPSET (OPTION,VALUE)
UQUERY	Obtains current value of specified variable in GSA
	CALL UQUERY (OPTION,VALUE)
UREAD	Allows alphanumeric input from the graphic terminal
	CALL UREAD (X,Y,DATA,COUNT,FLAG)

URECT Draws a rectangle

CALL URECT (X,Y)

UREPRO Reproduce contents of psuedo-display file on current display device

CALL UREPRO (FILENR, STATUS)

URESET Resets GSA variables to default conditions

CALL URESET

UROTAT Creates a user coordinate system at current position rotated as specified

CALL ROTAT (ANGLE)

USAREA Changes device boundaries to maintain a one to one aspect ratio with the current window boundaries

CALL USAREA

USAVE Saves all the variables of the Graphics Status Area

CALL USAVE (ARRAY)

USAXIS Draws a single axis in any of three coordinates

CALL USAXIS (AXIS,XSTART,YSTART,ZSTART,DIST)

USCALE Creates a user coordinate system at current position with specified scale

CALL USCALE (SX,SY)

USCATR Draws a scatter plot

CALL USCATR (X,Y,PTS)

USET Sets a graphics status area variable to a given value

CALL USET (OPTION)

USHOW Causes the named frame/segment of graphical information to be made visible

CALL USHOW (NAME)

USPLIN Fits a cubic spline interpolatory curve to the input data

CALL USPLIN (X,Y,PTS,RETX,RETY,RETPTS)

USTART Initializes the graphics status area

CALL USTART

USTRCT Defines the start of a graphic data structure

CALL USTRCT (NAME)

USTUD — Returns limits of two dimensional virtual and display surfaces

CALL USTUD (ARRAY)

USVPN — Saves all pen-related variables of the graphics status area

CALL USVPN (ARRAY)

USVTR — Saves coordinate system related variables in the graphics status area

CALL USVTR (ARRAY)

UTAXIS — Draws a time series axis with appropriate alphameric and numeric labels

CALL UTAXIS (BEGIN,PERIOD,YMIN,YMAX)

UTERM — Defines the end of a graphic data structure

CALL UTERM (NAME)

UTILTY — Performs data structure utility functions

CALL UTILTY (OPTION,VALUE)

UTSFIT — Fits an exponentially smoothed curve to time series data

CALL UTSFIT (ARRAY,POINTS,FCST,ALPHA)

UVIEW — Defines position of viewer in relation to environment, and the direction of view.

CALL UVIEW (XVIEW,YVIEW,ZVIEW,XSITE,YSITE,ZSITE)

UVWPLN — Defines the location of the view (projection) plane

CALL UVWPLN (DISTAN)

UWAIT — Waits a given number of seconds

CALL UWAIT (SECONDS)

UWHERE — Returns the coordinates of the current pen position in user units

CALL UWHERE (X,Y)

UWINDO — Sets the virtual window boundaries

CALL UWINDO (XMIN,XMAX,YMIN,YMAX)

UWLOOK — Adjusts both virtual window, and user display area to cover given portion of virtual space

CALL UWLOOK (XMIN,YMAX,YMIN,YMAX)

UWRITE — Prints information, then restores pen to location on input

CALL UWRITE (X,Y,DATA)

UWRIT1 Allows alphanumeric output at current position under one option

CALL UWRIT1 (DATA,OPTION)

UZWNDO Sets the hither/yon window boundaries for Z-clipping

CALL UZWNDO (ZMIN,ZMAX)

U3AXIS Creates a set of axes in 3 space

CALL U3AXIS (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX)

U3CALL Invokes an existing graphics data structure in 3 space.

CALL U3CALL (X,Y,Z,SX,SY,SZ,RX,RY,RZ,NAME)

U3CSYS Creates a new coordinate system in 3 space

CALL U3CSYS (X,Y,Z,SX,SY,SZ,RX,RY,RZ)

U3DRAW Draws a solid line in 3-D space

CALL U3DRAW(X,Y,Z)

U3GRIN Gets 3-D coordinates and a character from terminal and returns them

CALL U3GRIN (X,Y,Z,ICHAR)

U3IMAG Applies general 3-D image transformations to 'retained' frames/segments

CALL U3IMAG (X,Y,Z,SX,SY,SZ,RX,RY,RZ,SEGNAM)

U3LINE Connects 3-D arrays of points (X,Y,Z) with current line option

CALL U3LINE (X,Y,Z,PTS)

U3MOVE Moves pen invisibly in 3-D space

CALL U3MOVE (X,Y,Z)

U3PEN Draws a line from current pen position to given 3-D coordinates

CALL U3PEN (X,Y,Z)

U3PEN1 Sets one 'USET' option for this call only before executing 3-D pen movement

CALL U3PEN1(X,Y,Z,OPTION)

U3PLACE Applies 3-D translation image transformation to 'retained' frames/segments

CALL U3PLAC(X,Y,Z,SEGNAM)

U3PLOT Draws a general purpose graph in 3-D space

CALL U3PLOT (X,Y,Z,CURVES,PTS,OPTS)

U3PRNT Displays textual data at pen position in 3-D space

CALL U3PRNT (X,Y,Z,DATA)

U3ROTA Creats a user coordinate system in three space at current pen position, rotated as specified

CALL U3ROTA (RX,RY,RZ)

U3SCAL Creates a user coordinate system in three space at current per position, scaled as specified

CALL U3SCAL (SX,SY,SZ)

U3STUD Gives current setting of the 3-D user display area and windows

CALL U3STUD (ARRAY)

U3WHER Returns current pen position in current units in 3-D space

CALL U3WHER (X,Y,Z)

U3WNDO Sets the virtual 3-D window boundaries

CALL U3WNDO (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX)

U3WRIT Displays textual information in 3-D space, returns pen to original position

CALL U3WRIT (X,Y,Z,DATA)

APPENDIX B

USET OPTIONS

TABLE 1

USET OPTIONS MOST FREQUENTLY USED

TO REQUEST ABSOLUTE COORDINATE PLOTTING (DEFAULT):

USET ('ABSOLUTE') USET ('ABSO')
USET ('ABSb')

TO REQUEST RELATIVE OR INCREMENTAL COORDINATE PLOTTING:

USET ('RELATIVE') USET ('RELA')

TO REQUEST RECTANGULAR COORDINATES (DEFAULT):

USET ('RECTANGULAR') USET ('RECT')

TO REQUEST POLAR COORDINATES:

USET ('POLAR') USET ('POLA')

TO REQUEST ANGULAR VALUES IN DEGREES (DEFAULT):

USET ('DEGREES') USET ('DEGR')

TO REQUEST ANGULAR VALUES IN RADIANS:

USET ('RADIANS') USET ('RADI')

TO REQUEST PLOTTING IN VIRTUAL SPACE (DEFAULT):

USET ('VIRTUAL') USET ('VIRT')

TO REQUEST PLOTTING IN DEVICE SPACE:

USET ('DEVICE') USET ('DEVI')

TO REQUEST DEVICE SPACE PLOTTING UNITS IN INCHES (DEFAULT):

USET ('INCHES') USET ('INCH')

TO REQUEST DEVICE SPACE PLOTTING IN RASTER UNITS:

USET ('RASTERUNITS') USET ('RAST')

TO REQUEST DEVICE SPACE PLOTTING IN FONT (CHARACTER SPACE) UNITS:

```
USET ('FONTUNITS')        USET ('FONT')
```

TO REQUEST DEVICE SPACE PLOTTING IN PERCENT UNITS:

```
USET ('PERCENTUNITS')     USET ('PERC')
```

TO REQUEST PLOTTING OF A VISIBLE LINE (DEFAULT):

```
USET ('LINE')
```

TO REQUEST PLOTTING OF INVISIBLE LINES:

```
USET ('MOVE')
USET ('NOLINE')           USET ('NOLI')
USET ('NObLINE')          USET ('NObL')
```

TO REQUEST PLOTTING OF LINES WITH ARROW TERMINATORS:

```
USET ('ARROW')            USET ('ARRO')
```

TO REQUEST PLOTTING OF DASHED LINES:

```
USET ('DASH')
```

TO REQUEST PLOTTING OF TIC LINES:

```
USET ('TICLINE')          USET ('TICL')
```

TO REQUEST PLOTTING OF LINES WITH ONLY ENDPOINTS VISIBLE:

```
USET ('POINT')            USET ('POIN')
```

TO REQUEST PLOTTING OF LINES COMPOSED OF CHARACTERS:

```
USET ('ALPHANUMERICLINES')  USET ('ALPH')
```

TO REQUEST PLOTTING OF INVISIBLE LINES WITH CHARACTERS AT THEIR END-POINTS:

```
USET ('CHARACTER')        USET ('CHAR')
```

TO REQUEST CHARACTER OUTPUT IN THE FORM OF HARDWARE CHARACTERS (DEFAULT):

```
USET ('HARDWARE')         USET ('HARD')
```

TO REQUEST CHARACTER OUTPUT IN THE FORM OF SOFTWARE CHARACTERS:

USET ('SOFTWARE') USET ('SOFT')

TO REQUEST UPRINT/UWRITE OUTPUT IN TEXT FORMAT (DEFAULT):

USET ('TEXT')

TO REQUEST UPRINT/UWRITE OUTPUT IN INTEGER FORMAT:

USET ('INTEGER') USET ('INTE')

TO REQUEST UPRINT/UWRITE OUTPUT IN REAL FORMAT:

USET ('REALNUMBER') USET ('REAL')

TO REQUEST PLOTTING WITH RESPECT TO THE SYSTEM AXIS (DEFAULT):

USET ('SYSTEM') USET ('SYST')

TO REQUEST PLOTTING WITH RESPECT TO THE USER AXIS:

USET ('USERAXIS') USET ('USER')

TO REQUEST A LINE TYPE/CHARACTER TERMINATOR COMBINATION:

USET ('X$bb') USET ('X$')

Where

$ IS THE DESIRED CHARACTER TERMINATOR

AND

X, THE LINE TYPE SPECIFICATION, IS AS FOLLOWS:

A	ALPHANUMERIC LINE
D	DASHED LINE
L	SOLID LINE
N	NULL OR INVISIBLE LINE
T	TIC LINE
b	denotes blank

TABLE 2

USET OPTIONS BY ALPHABETICAL ORDER:
(Default Options are in Bold Type)

Option Name		To Request
'AARROW'		Alphanumeric lines with arrow terminators
'ABACKARROW'		Alphanumeric lines with back arrow terminators
'ABEND'	3D,*	To halt execution when error count exceeds certain limit
'ABORT'	3D	To halt execution when error count exceeds certain limit
'ABSb'		Plotting in an absolute coordinate system
'ABSOLUTE'		Plotting in an absolute coordinate system
'ABUTTING'		Abutting of display surface pages
'ACENTER'		To center character output about given location
'ACHARACTER'		Alphanumeric lines with character terminators
'ACOORDINATE'		Alphanumeric lines with ending coordinates indicated
'ADDITIVE'	*	Additive color blending mode
'ADOUBLEARROW'		Alphanumeric lines with arrowhead terminators
'ALLDISPLAYS'		Routing of graphical output to all devices
'ALPHANUMERIC'		Alphanumeric lines with no terminators
'ALTERNATEDISPLAY'		Routing of graphical output to first alternate device
'ANNUAL'		Time series axis scale in yearly intervals
'ANULL'		Alphanumeric lines with no terminators
'APOINT'		Alphanumeric lines with point terminators
'ARROWLINE'		Solid lines with arrow terminators
'ASYMBOL'		Alphanumeric lines with character terminators
'AUTOSCALE'		Automatic scaling for higher level graphing
'BACKARROWLINE'		Solid lines with back arrow terminators
'BALL'		Track ball graphical input
'BLACK'		Background color to be black
'BBLUE'		Background color to be blue
'BCYAN'		Background color to be cyan
'BESTFORMAT'		Numeric label output in best possible format
'BIHOURLY'		Time series axis scale in two hour intervals
'BLACK'		Switch to pen color black
'BLUE'		Switch to pen color blue
'BMAGENTA'		Background color to be magenta
'BRED'		Background color to be red
'BRIGHT'		Highest possible intensity for output
'BUILD'	3D	Structure building
'BWHITE'		Background color to be white
'BYELLOW'		Background color to be yellow
'CENTIMETERS'		Device space coordinates are in centimeters
'CHARACTER'		Null or invisible lines with character terminators
'CJUSTIFICATION'		Alphanumeric center justification
'COMPRESSED'*	3D	Data structure editing option
'CONTINUOUS'		Curved lines be interpreted as one pen operation
'COORDINATES'		UPRINT/UWRITE output in (X,Y) coordinate format
'CURSOR'		Graphic cursor as graphic input device
'CWINDOWING'	3D	Circular windowing
'CYAN'		Switch to pen color cyan
'CYLINDRICAL'		Coordinates are to be of the form (R,THETA,Z),

NOTE: b - is a blank or space
** - means not implemented*

		where R is the number of units of radius in the X,Y plane, THETA is the number of angular units around the Y axis, and Z is the number of units along the Z axis
'DAILY'		Time series axis scale in daily intervals
'DARROW'		Dashed lines with arrow terminators
'DASH'		Dashed lines with null terminations
'DATE'		Time series axis scale in date series interval
'DBACKARROW'		Dashed lines with arrow terminators
'DCHARACTER'		Dashed lines with character terminators
'DCOORDINATES'		Dashed lines with endpoint coordinates indicated
'DDOUBLEARROW'		Dashed lines with double arrow terminators
'DEFERRED'	*	Deferred error output until UEND is called
'DEGREES'		Angular information be interpreted in degrees
'DESENSITIZE'	I	Disabling of pick sensitivity
'DETECTABLE'		Enabling of pick sensitivity
'DEVICE'		Plotting in device space
'DIGITIZER'		Digitizer is graphics input device
'DIMb'		Lowest possible intensity for output
'DIMENSIONLINE'		Solid lines with arrow terminators
'DISPLAY'		Plotting in device space
'DNULL'		Dashed lines with no terminators
'DOUBLEARROW'		Solid lines with double arrow terminators
'DPOINT'		Dashed lines with point terminators
'DSYMBOL'		Dashed lines with character terminators
'DUMP'		To select dump option
'ECHO'*	I	Echo alphanumeric input option
'EDGEAXIS'		X and Y axis labels at edge of graph
'ERROROUTPUT'		Immediate error output
'EXECUTE'	3D	Execution of data structure commands as they are built
'EXPANDED'*	3D	Data structure editing option
'EXTENDEDMENU'*	I,3D	Extended menuing option
'EXTRALARGE'		Extra large character size
'FAST'		Fast blink rate
'FITLINEAR'		Fit linear function to plotted lines
'FITPOLYNOMIAL'		Fit least squares polynomial to plotted points
'FITSPLINE'		Fit cubic spline curve to plotted points
'FNUMBERMODE'		Frame identifiers provided as numbers
'FONTUNITS'		To indicate device space plotting in font units
'FULLSCALE'		Full scaling for higher level graphing
'FUNCTIONKEYS'		Function key is graphics input device
'GAPPED'		Alternate light and dark line output
'GFORMAT'		Numeric label output in FORTRAN real (E or F) format
'GOTHIC'		Gothic character font (standard GCS font)
'GRADS'	3D	Angular units to be measured in grads
'GREEN'		Switch to pen color green
'GRIDAXIS'		Grid axes for higher level graphing
'HARDWAREFONT'		Output of hardware generated characters
'HIGHLIGHTED'		Highlighted segments
'HITHER/YONbCLIPPING'		Z axis clipping
'HORIZONTAL'	3D	Alphanumeric output to be printed horizontally
'HOURLY'		Time series axis scale in twenty-four hourly intervals
'IFORMAT'		Numeric label output in integer format
'IGNORE'	3D	Ignore duplicate studies on merge file

NOTE: b - is a blank or space
** - means not implemented*

'INCHES'		Device space coordinates in inches
'INCREMENTAL'		Plotting in an incremental coordinate system
'INTEGER'		UPRINT/UWRITE output in integer format
'INVISIBLE'		Invisible construction of frames and segments
'ITALICS'		Italic character format
'JOYSTICK'	I	Joystick as graphic input device
'KEYBOARD'	I	Keyboard as (pseudo) graphic input device
'LARGE'		Large character size
'LARROW'		Solid lines with arrow terminators
'LBACKARROW'		Solid lines with back arrow terminators
'LCHARACTER'		Solid lines with character terminators
'LCOORDINATES'		Solid lines with endpoint coordinates indicated
'LDOUBLEARROW'		Solid lines with double arrow terminators
'LEFT'	3D	Left handed coordinate system
'LETTER'		UPRINT/UWRITE output in text format
'LIGHTPEN'	I	Light pen as graphic input device
'LINE'		Solid lines with null terminators
'LJUSTIFICATION'		Left justification of alphanumeric strings
'LNULL'		Solid lines with null terminators
'LOGARITHMIC'		Applies logarithmic transforms to all components
'LOGOBJECT'	3D	Logarithmic transforms are to be applied before any other transformations. (In this mode, log transforms may be applied to angle or radius components of 'CYLINDRICAL', 'POLAR', or 'SPHERICAL' coordinates)
'LOGORIGINALUNITS'		Application of log scaling before conversion to rectangular
'LOGSYSTEM'	3D	Logarithmic to be applied after conversion to 'SYSTEM' coordinates (in this mode, the logarithmic coordinate system axes.)
'LOGUSER'	3D	Logarithmic transforms are to be applied after conversion to 'ABSOLUTE'. 'RECTANGULAR', 'USER' coordinates but before conversion to 'SYSTEM' coordinates. (In this mode, the logarithmic scaling will be applied along the current 'USER' coordinate system axes.)
'LOGXAXIS'		Base ten log X axis drawing
'LOGYAXIS'		Base ten log Y axis drawing
'LOWERCASE'		Lower case to be 'TEXT' case
'LPOINT'		Solid lines with point terminators
'LSYMBOL'		Solid lines with character terminators
'MAGENTA'		Switch to pen color magenta
'MEDIUM'		Medium hardware character size
'MESSAGEDEVICE'		Alphanumeric I/O routed to a message device
'MILS'	3D	Angular units to be measured in mils
'MINUTELY'		Time series axis scale in minute intervals
'MONTHLY'		Time series axis scale in monthly intervals
'MOUSE'	I	Analog mouse as graphic input device
'MOVE'		Invisible lines with null terminators
'MULTIPLE'	3D	Multiple data structure invocation
'NARROW'		Invisible lines with arrow terminators
'NBACKARROW'		Invisible lines with back arrow terminators
'NCHARACTER'		Invisible lines with character terminators
'NCOORDINATES'		Invisible lines with double arrow terminators
'NDOUBLEARROW'		Invisible lines with double arrow terminators
'NEGATIVESIDE'	3D	Labels will be to the left or below the axes
'NEWSCALE'		New scale for higher level graphing
'NNULL'		Invisible lines with null terminators

NOTE: b - is a blank or space
** - means not implemented*

'NObLINE'		Invisible lines with null terminators
'NOABORT'	3D	Do not terminate if error count exceeds specified limit
'NOAXES'		No axes be drawn for higher level graphing
'NOBLINK'		No blinking to occur
'NOBUILD'	3D	No structure building
'NOCENTER'		Text output starts at given point
'NODUMP'	3D	No dump performed
'NOECHO'	I,3D	No echoing of alphanumeric input
'NOEXECUTE'	3D	No execution of data structure commands as they are built
'NOFIT'		No curve fitting for higher level graphics
'NOHIGHLIGHTING'		No segment highlighting
'NOITALICS'		No slanting of software characters
'NOLINE'		Invisible lines with null terminators
'NOLOGARITHMS'	3D	No logorithmic transform application
'NONUNIFORM'	3D	High level plotting option
'NOMARK'		Invisible marks with null terminators
'NONABUTTING'		No abutting of display surface pages
'NONRETAINEDSEGMENTS'		Create segments in non-retained form
'NONUNIFORM'		Nonuniform scaling of higher level grading
'NOORIGIN'	3D	No origin to be forced for 'AUTOSCALE' or 'FULLSCALE' scaling options
'NOREPEAT'	3D	No coordinate repeating for high level plotting
'NOREWIND'		No rewind of structure save files
'NORMALINTENSITY'		Normal intensity for output
'NOSCRIPT'	3D	To disable any superscripting or subscripting of text output
'NOSIGNIFICANTZEROES'		Suppression of display of significant zeros
'NOSUPERSCRIPT'	3D	Same as 'NOSCRIPTING'
'NOTRAIL'	3D	Record of which GCS routines are invoked is not listed
'NOWINDOWING'	3D	Disable GCS windowing routine
'NOXLABEL'		No labels are to be drawn for graphing
'NOXREPEAT'	3D	To indicate that a component is provided for every X value in every curve in higher level graphing
'NOYLABEL'		No Y labeling for graphing
'NOYREPEAT'	3D	To indicate that a component is provided for every Y value in every curve in higher level graphing
'NOZCLIPPING'		No hither/yon clipping
'NOZLABELS'	3D	To indicate that no Z labels are to be drawn for higher level graphing
'NOZREPEAT'	3D	To indicate that a component is provided for every Z value in every curve in higher level graphing
'NObLINE'		Invisible line with null terminator
'NObMARK'		Line type
'NPOINT'		Invisible lines with point terminators
'NSYMBOL'		Invisible lines with character terminators
'OLDSCALE'		Old scale to be used for higher level graphing
'ORIGIN'	3D	An origin to be forced for 'AUTOSCALE' and 'FULLSCALE' scaling options
'ORTHOGRAPHIC'	3D	To specify orthographic projection in which the projection is parallel from all points
'OWNSCALE'		Own scale option for higher level graphing
'PARALLELLABELS'	3D	To specify that the main axis of the numeric labels will be parallel to the axis
'PENAXIS'		Axis intersection at current position for graphing

NOTE: b - is a blank or space
** - means not implemented*

'PENDOWN'		Solid lines with null terminators
'PENORIGIN'	3D	To force the current pen position to be included in the axis range.
'PENUP'		Invisible lines with null terminators
'PERIODIC'		Time series axis scale in accounting period intervals
'PERCENTUNITS'		Device space coordinates specified in percent units
'PERPENDICULARLABELS'	3D	To specify that the major axis the numeric labels will be perpendicular to the axis
'PERSPECTIVE'	3D	To specify perspective projection in which line length diminishes as the distances from the viewing position become greater
'PIRADIANS'	3D	Angular information be interpreted in Pi radians
'PLAINAXIS'		Plain axes to be drawn for high level graphing
'PLOTDEVICE'		Alphanumeric I/O to be directed to the plotting device
'POINT'		Invisible lines with point terminators
'POLAR'		Plotting in polar (RHO, THETA) units
'POSITIVESIDE'	3D	Labels will be above or to the right of the axis
'PRIMARYDEVICE'		Routing of graphical output to primary graphics device
'QUARTERLY'		Time series axis scale in quarter year intervals
'RADIANS'		Angular information be interpreted in radians
'RASTERUNITS'		To indicate device space coordinates are specified as is in raster units
'REAL'		UPRINT/UWRITE output in real number format
'RECTANGULAR'		Plotting on the user's reference axis
'REDb'		Switch to pen color red
'REFERENCE'		Plotting on the user's reference axis
'REFRESHEDSEGMENT'		Segments to be retained
'RELb'		Plotting in a relative coordinate system
'RELATIVE'		Plotting in a relative coordinate system
'REPLACE'	3D	Data structure building option
'RETAINEDSEGMENTS'		Segments to be retained structures from merge file
'REWIND'	3D	Data structure file handling command
'RIGHTHAND'	3D	Right handed coordinate system
'RJUSTIFICATION'		Right justification of alphanumeric character string
'RWINDOWING'	3D	Rectangular windowing
'SECONDLY'		Time series axis scale in second intervals
'SECRET'		Security classification secret
'SEGMENTED'		Curved lines be interpreted as multiple pen operations
'SEMIANNUAL'		Time series axis scale in semi annual intervals
'SENSITIZE'	3D	Make graphic segments visible
'SIGNIFICANTZEROES'		Display of significant zero
'SIMULATED HARDWARE CHARACTERS'		Output of simulated hardware characters
'SINGLE'	3D	Data structure invocation option
'SITEPOINT'	3D	Viewpoint distance to be measured from the view site
'SLOWBLINK'		Slow blink rate
'SMALL'		Use smallest hardware character size
'SOFTWAREFONT'		Output of software generated characters
'SONICPEN'	I	Sonic pen is graphics input device
'SPECIFIC'		To specify particular device units instead of percent units

NOTE: b - is a blank or space
** - means not implemented*

'SPHERICAL'		Coordinates are of the form (R,THETA,PHI) where R is the number of units of radius, THETA is the number of angular units around the Z axis, and PHI is the number of angular units around the X axis
'STANDARDMENU'	I,3D	Menuing option
'SUBSCRIPT'	3D	To specify that the output will be lowered from the specified line of text
'SUPERSCRIPT'	3D	To specify that the output will be raised from the specified line of text.
'SUPPRESSERRORS'		Error output be suppressed
'SYMBOL'		Invisible lines with character terminators
'SYSTEMAXIS'		Plotting on the system axis
'TABLET'	I	Analog tablet as graphic input device
'TARROW'		Tic lines with arrow terminators
'TBACKARROW'		Tic lines with back arrow terminators
'TCHARACTER'		Tic lines with character terminators
'TCOORDINATES'		Tic lines with endpoint coordinates indicated
'TDOUBLEARROW'		Tic lines with double arrow terminators
'TEXT'		UPRINT/UWRITE output in text format
'SUBTRACTIVE'*		Subtractive color blending mode
'TICAXES'		Tic axes to be drawn for higher level graphing
'TICLINE'		Tic lines with null terminators
'TNULL'		Tic lines with null terminators
'THINLINES'		Line width to be thin
'TOPSECRET'		Security Classification Top Secret
'TPOINT'		Tic lines with point terminators
'TRAIL'	3D	To indicate by an identification number which GCS routine is involved.
'TSYMBOL'		Tic lines with character terminators
'TWELVEHOUR'		Time series axis scale in twelve hour intervals
'TWENTYFOURHOUR'		Time series axis scale in twenty four-hour intervals
'UNCLASSIFIED'		Security classification unclassified
'UNDETECTABLE'		Disabling pick sensitivity
'UNIFORM'*	3D	High level graphing system
'UNINTERRUPTED'		Non-gapped line output
'UPPERCASE'		Upper case to be 'TEXT' case
'USER'		Plotting on a user defined axis system
'VERTICAL'	3D	Alphanumeric output to be spaced vertically
'VIEWPOINT'	3D	View port distance to be measured from the view point
'VIRTUAL'		Plotting in virtual space
'VISIBLE'		Visible framed output
'WEEKLY'		Time series axis scale in weekly intervals
'WHITE'		Switch to pen color white
'WIDELINES'		Line width to be wide
'WORKINGAXIS'		Plotting on the user's working (temporary axis)
'WORLDCOORDINATESYSTEM'		Coordinate system to be the default axis system
'XABSOLUTE'	3D	To specify the X coordinates with respect to the origin of the current coordinate system for indicated components. Y and Z components are to be specified with respect to the current beam/pen position
'XALPHANUMERIC'		The X axis will have an alphanumeric label
'XAXIS'		The X axis be drawn for high level graphing
'XBOTHLABELS'		The X axis will have both alphanumeric and numeric labels

NOTE: b - is a blank or space
** - means not implemented*

'XCONSTANT'	3D	To indicate that the X component does not vary during the drawing of any curve in higher level graphics
'XEDGEYZEROAXIS'		The X axis at edge of graph
'XLOGARITHMIC'		Logarithmic X and linear Y plotting
'XNEGATIVE'	3D	Negative X axis represents up in 3D graphics
'XNUMERIC'		An X axis numeric label
'XPOSITIVE'	3D	Positive X axis represents up in 3D graphics
'XRELATIVE'	3D	To specify the X coordinates with respect to the current beam/pen position. Y and Z components are to be specified with respect to the origin of the current coordinate system
'XREPEAT'	3D	To indicate that one set of X valves is provided which will be reused for every curve in higher level graphing
'XYAXES'		The X and Y axes be drawn for high level graphing
'XYABSOLUTE'	3D	To specify the X and Y coordinates with respect to the origin of the current coordinate system for the indicated components. Z components are to be specified with respect to the current beam/pen position
'XYCOORDINATES'		To specify that the data printed by UPRINT/ UWRITE is in the form of an (X,Y) coordinate pair.
'XYLOGARITHMIC'	3D	Logarthmic X and Y plotting, linear Z plotting
'XYPLANE'	3D	Labels are to be drawn in the plane formed by the X and Y axes
'XYRELATIVE'	3D	To specify the X and Y components with respect to the current beam/pen position. Z components are to be specified with respect to the origin of the current coordinate system
'XYVIEW'	3D	To view plane formed by X and Y axes
'XYZAXES'	3D	All three axes are to be drawn for higher level graphing
'XYZCOORDINATES'	3D	Text printed by U3PRNT/U3WRIT to be in form of (X,Y,Z) triplet
'XYZLOG'	3D	Applies logamithmic transforms to all three components
'XYZVIEW'	3D	To view all three axes
'XYZb'	3D	To set the rotation application order as indicated
'XYCOORDINATES'		UPRINT/UWRITE output in (X,Y) coordinate format
'XZABSOLUTE'	3D	To specify the X and Z coordinates with respect to the origin of the current coordinate system for the indicated components. Y components are to be specified with respect to the current beam position
'XZAXES'	3D	The X and Z axes are to be drawn for higher level graphing
'XZEROYEDGEAXIS'		The X axis adjacent to boundary of display area
'XZLOGARITHMIC'	3D	Applies Logarithmic transformation to X and Z components
'XZPLANE'	3D	Label plane to be plane formed by X and Z axis
'XZRELATIVE'	3D	To specify the X and Z components with respect to the current beam/pen position. Z components are to be specified with respect to the origin of the current coordinate system
'XZVIEW'	3D	To view the plane formed by the X and Z axes
'XZYb'	3D	To set the rotation application order as indicated

NOTE: b - is a blank or space
** - means not implemented*

'YABSOLUTE'	3D	Y coordinates are specified with respect to the origin of the current coordinate system for the indicated units. X and Z components are specified with respect to the current beam/pen position
'YALPHANUMERIC'		Y axis alphabetic label
'YAXIS'		The Y axis to be drawn for high level graphing
'YBOTHLABELS'		Y axis having alphabetic and numeric labels
'YCONSTANT'	3D	To indicate that the Y component does not vary during the drawing of any curve
'YEARLY'		Time series axis scale in yearly intervals
'YEDGEXZEROAXIS'		The Y axis at edge of graph
'YELLOW'		Switch to pen color yellow
'YLOGARITHMIC'		Logarithmic Y plotting
'YNEGATIVE'	3D	Negative Y direction represents up in 3D graphics
'YNUMERIC'		Y axis numeric label
'YPOSITIVE'	3D	Positive Y direction represents up in 3D graphics
'YRELATIVE'	3D	Y coordinates are specified with respect to the current beam/pen position for the indicated components X and Z components are to be specified with respect to the origin of the current coordinate system
'YREPEAT'	3D	To indicate that one set of Y valves is provided which will be reused for every curve in higher level graphing
'YXZb'	3D	To set the rotation application order as indicated
'YZABSOLUTE'	3D	Y and Z coordinates are specified with respect to the origin of the current coordinate system for the indicated under X components are specified with respect to the current beam/pen position
'YZAXES'	3D	The Y and Z axes to be drawn for higher level graphing
'YZPLANE'	3D	Label plane to be plane formed by Y and Z axes
'YZEROXEDGEAXIS'		The Y axis adjacent to boundary of display area
'YZLOGARITHMIC'	3D	Applies logarithmic transformations to Y and Z components
'YZRELATIVE'	3D	Y and Z coordinates to be specified with respect to the current beam/pen position for the Y and Z components, and the X component to be specified with respect to the current beam/pen position for the Y and Z components, and the X component to be specified with respect to the origin of the current coordinate system
'YZVIEW'	3D	To view the plane formed by the Y and Z axes
'YZXb'	3D	To set the rotation application order as specified
'ZABSOLUTE'	3D	Z coordinates to be specified with respect to the origin of the current coordinate system for the indicated component. X and Y components are to be specified with respect to the current beam/pen position
'ZALPHANUMERIC'	3D	Alphanumeric labels to be drawn for higher level drawing
'ZAXIS'	3D	Z axis is to be drawn for higher level graphing
'ZBOTHLABELS'	3D	Both numeric and alphanumeric labels to be drawn for higher level drawing
'ZCLIP'	3D	Clip in Z direction
'ZCONSTANT'	3D	To indicate that the Z component does not vary during the drawing of any curve
'ZEROAXES'		X and Y axes adjacent to boundary of display area

NOTE: b - is a blank or space
** - means not implemented*

'ZLOGARITHMIC'	3D	Applies logarithmic transform to Z component
'ZNEGATIVE'	3D	Negative Z axis represents up direction in 3D graphics
'ZNUMERIC'	3D	Numeric Z labels to be drawn for high level graphing
'ZPOSITIVE'	3D	Positive Z axis represents up direction in 3D graphics
'ZRELATIVE'	3D	Z coordinates are to be specified with respect to the current beam/pen position. X and Y components are to be specified with respect to the origin of the current coordinate system
'ZREPEAT'	3D	To indicate that one set of Z values is provided which will be reused for every curve in higher level graphing
'ZXYb'	3D	To set the rotation application order as indicated
'ZYXb'	3D	To set the rotation application order as indicated
'12HOUR'		Twelve hour time axis
'13WEEK'		Thirteen week time axis
'2DCOORDINATES'	3D	To specify A coordinate terminator in which two components are listed. (This option is independent of the text coordinate options of 'XYCOORDINATES' and 'XYZCOORDINATES')
'24HOUR'		Twenty four time axis
'3DCOORDINATE'	3D	To specify a coordinate terminator in which all three components are listed. (This option is independent of the text coordinate option of 'XYCOORDINATES' and 'XYZCOORDINATES')

NOTE: b - is a blank or space
** - means not implemented*

APPENDIX C

UPSET OPTIONS

Option Name	Value
'ANGLE OF TEXT'	Is an angular value which specifies the angle of the text string in relation of the current X axis. Default value is 0.
'ATTENTION QUEUE SIZE'*	An integer indicating the number of words provided in the attention queue. Default is 0.
'ATTITUDE'	Is an angular value which specifies the orientation of the up direction axis with the sides of the window. Default value is 0. Meaning that the up direction is parallel whth the left and right window boundaries and pointing towards the top of the window.
'BACKGROUND COLOR'	Is an integer which specifies the color index within the color table for colored backgrounds. Values 0-7 are predefined to represent black, white, red, green, yellow, blue, magenta, and cyan, respectively. Default color is black or none.
'BASE OF LOGARITHMS'	Is a positive value which specifies the base of the logarithms used to perform logarithmic scaling or the character string 1HE, denoting the base of the Naperean logarithms. Default base is 10. This option sets the specified base along each coordinate component.
'BRIGHTNESS'	Is a value between 0 and 100. percent indicating the position in the range of possible line intensity settings from dimmest to brightest. Default brightness is 60%.
'CHARACTER'	Is a Hollerith character which becomes the current system character. The default system character is a star or asterisk (*).
'COLOR'	Is an integer which specifies the color index within the color table. Values 0-7 are predefined to represent black, white, red, green, yellow, blue, magenta, and cyan, respectively. Default color is device-dependent.
'COPY DELAY'	Is a value which indicates the number of seconds of delay required during generation of a hard copy. Default value is device-dependent and is preset to the value required by the selected device.
'DISTANCE'	Is a value measured from the current view plane distance base specifying the position of the view (projection) plane. The default is 0. measured from the view site.

*means not implemented

'FNT FILE'	Is a Fortran file number representing the file containing the font descriptors. Default is zero indicating no font file specified.
'FONT NAME'*	Is a Hollerith string indicating the desired character font. Default is 'GCS' which is the most efficient font.
'GREYSCALE'	Is a value indicating a particular grey level for terminals which support multiple grey scales rather than colors.
'GRID SPECIFICATION'	Is a value containing a dash specification to be used when generating grid axes. Default is 0. which indicates a solid line should be used.
'HORIZONTAL SIZE'	Indicates the width of a software character position in current user units. Default is 5 virtual units.
'INPUT FILE'	Is a Fortran file number indicating which file will be used to obtain graphics input. The default is set to the appropriate computer-system dependent file.
'LABELbROTATION'	Is the number of angular units the axes lbaels are to be rotated around the axes.
'LIBRARY FILE'	Is a Fortran file number indicating which file should be used by the GCS structure and segmentation facilities as a random work file. Default is 0. indicating no file has been provided.
'LOWER'	Is a Hollerith character which will be used by GCS as the indication to shift to lower case. Default character is '>'.
'MARKER INDEX'	Is an integer value selecting a marker/symbol. Default marker symbol is 0. indicating a point.
'ORIENTATION'	Is an angular value indicating the display orientation of GCS created software symbols and figures such as software characters, polygons, and rectangles. Default orientation is 0.
'OUTPUT FILE'	Is a Fortran file number indicating which file will be used for sending graphics output to the display device. The default is set to the appropriate computer system dependent file.
'POLYNOMIAL DEGREE'	Specifies the degree of the polynomial to be created in calculating a least squares fit through a collection of points. Default value is 5.
'PRECISION'	Specifies the number of significant digits to appear when displaying real numbers. Default value is 4.

*means not implemented

'READFILE'	Is a Fortran file number indicating which file will be used to obtain non-graphic input. The default is set to the appropriate computer system dependent file.
'ROTATION'	Same as 'ORIENTATION'.
'SCALEFACTOR'	Specifies a scale to be applied to GCS-created geometric figures such as polygons and rectangles.
'SCRIPTLEVEL'	Is an integer value indicating the scripting level to be set for textual output. Default value is 1.
'SETDASH'	Specifies the characteristics of the dashed lines to be plotted by UPEN. Default value is 56.
'SIZE'	Is a positive integer value which sets hardware character sizes. Values of 1 through 4 correspond to USET options 'SMALL', 'MEDIUM', 'LARGE', and 'EXTRA LARGE' respectively. Default value is 1 for 'SMALL' characters.
'SLANTANGLE'	Is an angular value indicating the amount of slant from the vertical for italicized software characters. Default value is approximately 18 degrees.
'SPAN ANGLE'	Is an angular value indicating the portion of a circle to be occupied by the pie chart. Default value is 360 degrees.
'SPECIFICATION UNITS'	Is a positive value indicating the number of specification units contained in device space for all directions. Default value is 1000.
'SPEED'	Specifies the speed of the communication line in characters per second. Default value is system dependent.
'START ANGLE'	Is an angular value indicating the starting position of the first wedge of the pie chart. Default value is 0.
'STRUCTURE TABLE SIZE'	Is an integer value indicating the number of words in the user provided structure table. Default value is 100
'SUBSCRIPT CHARACTER'	Is a Hollerith character which will be used to decrease the scripting level by 1. Default subscript character is 'the display code is a pound sign'.
'SUPERSCRIPT CHARACTER'	Is a Hollerith character which will be used to increase the scripting level by 1. Default superscript character is 'the display code is an underline sign'.

*means not implemented

'SZMARKER'	Is a value in current device units which specifies the size of software generated markers. Default value is device-dependent.
'TABHORIZONTAL'	Is an array of 10 elements containing 10 tab positions in current device units. Default value has all tab stops set to zero.
'TABVERTICAL'	Is an array of 10 elements containing 10 vertical tab positions in current device units. Default value has all tab stops set to zero.
'TERMINATOR'	Is a Hollerith character which will be used as the GCS string terminator character. Default character is a backslash.
'TICINTERVAL' TICLENGTH'	Specifies the distance in current user units between tic marks of a UPEN created tic line. Default value is 10.
'TICMINUS'	Specifies in current units the size of that portion of a tic mark which lies on the clockwise side of the tic line. Default value is .05 inches.
'TICPLUS'	Specifies in current units the size of that portion of a tic mark which lies onthe clockwise side of the tic line. Default value is .05 inches.
'TICX'	Specifies the distance between tic marks or grid lines along X axes. Default value is 0. indicating that a 'nice' number should be chosen.
'TICY'	Is the same as 'TICX' for Y axes.
'TICZ'	Is the same as 'TICX' for Z axes
'UPPER'	Is a Hollerith character which will be used by GCS as the indication to shfit to upper case. Default character is '<'.
'VERTICAL SIZE'	Indicate the height of a software character position in current user units. Default value is seven virtual units.
'WIDTH'	Is a value in current units of the width of a line. Default value is 0. indicating a thin line.
'WRITE FILE'	Is a Fortran file number indicating which file will be used to generate non-graphic output. The default is set to the appropriate computer system dependent file.
'XBASE OF LOGS'	Is a positive value which specifies the base for logarithmic scaling along the X component. Default value is 10.
'XLABEL'	Specifies the alphanumeric label to be displayed along X axes. Default value is 'X '.

*means not implemented

'XPERCENT'	Is a value specifying the portion of the width of a software character position to be occupied by a character. Default value is .65 indicating 65% of the width.
'XROTATION'	Is an angular value indicating the amount of rotation around the X axis for UINVOK structure invocations. Default value is 0.
'XSCALE'	Is the scale factor to be applied along the X axis for UNIVOK structure invocations. Default value is 1.
'XSIZE'	Is the size of hardware or simulated hardware character positions in current device units. Default value is device=dependent and corresponds to 'SMALL' hardware character size.
'XSPECIFICATION UNITS'	Is a positive value indicating the number of X specification units in device space. Default value is 1000.
'YBASE OF LOGS'	Is a positive value indicating the number of X specification units in device space. Default value is 1000.
'YBASE OF LOGS'	Is a positive value which specifies the base for logarithmic scaling along the Y component. Default value is 10.
'YLABEL'	Specifies the alphanumeric label to be displayed along the Y axis. Default value is 'Y '.
'YPERCENT'	Is a value specifying the portion of the height of a software character position to be occupied by a character. Default value is .65 indicating 65% of the height.
'YROTATION'	Is an angular value indicating the amount of rotation around the Y axis for UNIVOK structure invocations. Default value is 0.
'YSCALE'	Is the scale factor to be applied along the Y axis for UNIVOK structure invocations. Default value is 1.
'YSIZE'	Is the size of hardware or simulated hardware character positions in current device units. Default value is device-dependent and corresponds to 'SMALL' hardware character size.
'YSPECIFICATION UNITS'	Is a positive value indicating the number of Y specification units in device space. Default value is 1000.
'ZBASE OF LOGS'	Is a positive value which specifies the base for logarithmic scaling along the Z component. Default value is 10.

*means not implemented

'ZLABEL'	Specifies the alphanumeric label to be displayed along the Z axis. Default value is 'Z '.
'ZROTATION'	Is an angular value indicating the amount of rotation around the Z axis for UINVOK structure invocations. Default value is 0.
'ZSCALE'	Is the scale factor to be applied along the Z axis for UINVOK structure invocations. Default value is 1.

*means not implemented

APPENDIX D

GCS DEFAULT CONDITIONS

This appendix addresses those options which are present in the Graphics Status Area as default options. After a call to Subroutine USTART, the Graphics Compatibility System is set to the default conditions, as indicated. The default options can be divided into two groups: Basic Plotting Options and High Level Plotting Options.

I. Default Basic Plotting Options

Plotting is done in 'RECTANGULAR' and 'ABSOLUTE' coordinates on the 'SYSTEM' coordinate axis. The 'USER' coordinate axes are identical to the system axis. Plotting is done in 'VIRTUAL' space with a virtual window whose limits are from 0.0 to 100.0 in the X direction, and from 0.0 to 100.0 in the Y direction. The virtual window is mapped into a display area which is the largest square area on the device display surface. The right hand edge of the square corresponds to the right edge of the display surface, and the top edge of the square corresponds to the top edge of the display surface.

Character output in GCS will be, by default, in 'HARDWARE' character format, of type 'GOTHIC' and of 'MEDIUM' size. If 'SOFTWARE' characters are requested via USET then the default horizontal size is five (5.0) virtual units, and the default vertical size is seven (7.0) virtual units.

By default, angular units for angular specifications are in 'DEGREES'. If the user switches to 'DEVICE' space, then all length or distance units (i.e. rectangular coordinates) are in terms of 'INCHES'. The default alphanumeric margins are the boundaries of the plotting surface. For alphanumeric I/O through GCS (i.e. from UPRINT or UREAD), the data will be assumed to be in 'TEXT' mode. The alphanumeric output defaults to the 'PLOT' device if possible, and any graphic output will go to all devices in a cluster. Graphic input is from the primary input device; the number of digits of precision for numeric output is four (4); and GCS detected errors are signalled as they occur.

A line which is drawn by a subroutine in GCS is considered to consist of two parts; a line type and a line terminator. The line type may be solid (visible), ticked, null (invisible), dashed, or alphanumeric. The line may be terminated by an arrow, a back arrow, double arrows, a character, a point, a symbol, a set of coordinate values, or nothing at all. The default line type in GCS is 'SOLID' with 'NULL' termintors ('LNULL'). If 'TICLINES' are requested via USET option, then the default tic interval is ten (10.0) virtual units. It is the user's responsibility to insure that the tic interval is appropriate if he switches to 'DEVICE' space or alters the default virtual window setting or the default display area setting. If 'DASHLINES' are requested, then the default dash specification (56.) will result in a dashed line which is alternately light and dark, in icrements of approximately 0.075 inches. If a 'CHARACTER' line terminator is requested but no character is specified by way of Subroutine UPSET, then the character asterisk (*) is used. Similarly, the asterisk is used to compose the line type if 'ALPHANUMERIC' lines are requested.

For 3D operations, the viewing environment is set up to simulate a 2D only environment. The view point is located at (0.,0.,150.), the view site is at (0.,0.,0.) and the view plane is located at the view site. The system coordinate system is considered to be 'RIGHT-HANDED' with the 'ZPOSITIVEAXIS' representing up. Note that since the view is down the Z axis, the up direction degenerates to 'YPOSITIVE'. The Z axis clipping planes are 150. (hither plane) and 1.0E+30 (yon plane) with 'NOZCLIPPING'.

II. Default High Level Plotting Options

For each call to UPLOT or UPLOT1, the data values which represent the curves are examined, and a 'NEWSCALE' is created. UAXIS will be invoked to create 'XYAXES'. The data values will be examined, appropriate limits for the established. Numeric labels only will be output for the X axis and the Y axis. The axes will be positioned at the 'EDGE' of the plot. Both the X axis and the Y axis will be drawn in a linear coordinate space. The axis lines will be ticked. If a time series axis is plotted by invoking Subroutine UTAXIS, the default interval for the X axis will be 'DATE'. No curves will be fit to the data values, but if 'FITPOLYNOMIAL' is requested, then subroutine UPLOT will attempt to fit a fifth degree poynomial to the data.

III. Summary

COORDINATE SPACE:	'VIRTUAL'
COORDINATE TYPE:	'ABSOLUTE'
COORDINATE SYSTEM:	'RECTANGULAR'
COORDINATE AXIS:	'SYSTEM'
VIRTUAL WINDOW:	0.0 TO 100.0 X DIRECTION 0.0 TO 100.0 Y DIRECTION
DISPLAY AREA:	Largest square area which is right-up on the display surface of the device.
DEVICE SPACE UNITS:	'INCHES'
ANGULAR UNITS:	'DEGREES'
LINE TYPE:	'LINE' or 'LNULL'
SYSTEM CHARACTER:	asterisk (*)
TIC INTERVAL:	10.0 virtual units
DASH SPECIFICATION:	56.
CHARACTER TYPE:	'HARDWARE'
CHARACTER FONT:	'GOTHIC'
CHARACTER SIZE:	'MEDIUM'
SOFTWARE CHARACTER SIZE:	5.0 virtual units horizontal 7.0 virtual units vertical
INPUT/OUTPUT FORMAT:	'TEXT'
ALPHANUMERIC MARGINS:	Device display surface boundaries.
OUTPUT ROUTE:	'PLOTDEVICE'
OUTPUT DISTRIBUTION:	'ALLDEVICES'
GRAPHIC INPUT:	Primary input device
DIGITS OF PRECISION:	4
ERROR HANDLING:	'IMMEDIATE OUTPUT'
AXIS SCALING:	'AUTOSCALE'
AXIS LABELING:	'XNUMERICLABEL' 'YNUMERICLABEL'
AXIS POSITIONING:	'EDGEAXIS'
AXIS TYPE:	'TICAXIS'
AXIS EXISTENCE:	'XYAXES'
AXIS COORDINATES:	'NOLOGARITHMS'
TIME SERIES AXIS SCALE:	'DATE'

APPENDIX E

USET OPTIONS BY CLASS

Coordinate Mode

'ABSOLUTE'	'INCREMENTAL'	'RIGHTHANDED'	'ZABSOLUTE'
'RELATIVE'	'XYRELATIVE'	'YABSOLUTE'	'ZRELATIVE'
'XABSOLUTE'	'XZABSOLUTE'	'YRELATIVE'	
'XRELATIVE'	'XZRELATIVE'	'YZABSOLUTE'	
'XYABSOLUTE'	'LEFTHANDED'	'YZRELATIVE'	

Coordinate Type

'RECTANGULAR'	'CYLINDRICAL'	'LOGOBJECT'	'XYZLOGARITHMIC'
'POLAR'	'SPHERICAL'	'LOGSYSTEM'	'XZLOGARITHMIC'
'LOGARITHIMC'	'XLOGARITHMIC'	'LOGUSER'	'YZLOGARITHMIC'
'YLOGARITHMIC'	'XYLOGARITHMIC'	'NOLOGARITHMS'	'ZLOGARITHMIC'

Coordinate Space

'VIRTUAL'	'SPECIFIC'
'DEVICE'	'DISPLAY'

Device Space Units

'INCHES'
'CENTIMETERS'
'FONTUNITS'
'PERCENTUNITS'
'RASTERUNITS'

Angular Units

'DEGREES'	'GRADS'
'RADIANS'	'MILS'
'PIRADIANS'	

Frame Composition

'INVISIBLE'
'VISIBLE'

Line Type

'LINE'	'LNULL'	'NOLINE'	'LBACKARROW'
'DASH'	'DNULL'	'NOMARK'	'LCOORDINATE'
'POINT'	'NPOI'	'NNULL'	'NO LINE'
'TICLINE'	'TNULL'	'LARROW'	'NO MARK'
'CHARACTER'	'BACKARROWLINE'	'LDOUBLEARROW'	'DIMENSIONLINE'
'SYMBOL'	'COORDINATELINE'	'MOVE'	'NSYMBOL'
'ALPHANUMERIC'	'NCHA'	'ARROWLINE'	'ANULL'
'DOUBLEARROWLINE'			

Lines that are drawn by GCS are composed on a line type and a line terminator. A large number of line types and terminators are possible. The specification is composed by

combining the first letter of the line type with the name of the terminator. The following tables illustrate the possible linetypes and terminators:

LINE TYPE	**FIRST LETTER**
LINE	L
DASH	D
ALPHANUMERIC	A
NULL (INVISIBLE)	N
TICLINE	T

Line Terminators

NULL (NO TERMINATORS)
CHARACTER
SYMBOL
COORDINATE
POINT
ARROW
BACKARROW
DOUBLEARROW

Line Repeatability

'UNINTERRUPTED'
'GAPPED'

Curve Approximation

'CONTINUOUS'
'SEGMENTED'

Blink Rate

'NOBLINK'
'FASTBLINK'
'SLOWBLINK'

Color

'WHITE'	'BLUE'
'BLACK'	'RED'
'GREEN'	'MAGENTA'
'CYAN'	'YELLOW'

Intensity

'DIM'
'BRIGHT'

Coordinate Axis

'SYSTEMAXIS'
'USERAXIS'

Coordinate Axis Composition

'WORKINGAXIS'
'REFERENCEAXIS'

Character Format

'GOTHIC'
'UPPERCASE'
'ITALIC'
'LOWERCASE'

Character Size

'SMALL'
'MEDIUM'
'LARGE'
'EXTRALARGE'

Device Routing

'PLOTDEVICE'
'MESSAGEDEVICE'

Device Selection

'ALLDEVICES'
'PRIMARYDEVICE'
'ALTERNATEDEVICE'

Graphic Input

'CURSORS'	'NOECHO'
'KEYBOARD'	'ECHO'
'LIGHTPEN'	'ORMODE'
'JOYSTICK'	'ANDMODE'
'BALL'	'PROCEED'
'MOUSE'	'WAIT'

Character Type

'HARDWARE'
'SOFTWARE'

Error Conditions

'ERROR OUTPUT'	'DUMP'	'NOABORT'	'TRAIL'
'SUPPRESSERRORS'	'ABEND'	'NODUMP'	
'DEFERERRORS'	'ABORT'	'NOTRAIL'	

Structure Definition

'BUILD'
'NOBUILD'

Structure Building

'EXECUTE'
'NOEXECUTE'

Structure File Manipulation

'APPEND'	'REPLACE'

'IGNORE' 'REWIND'
'MULTIPLE' 'SINGLE'

Structure Editing

'COMPRESSED'
'EXPANDED'

Axis Scaling

'AUTOSCALE'
'FULLSCALE'
'OWNSCALE'

Axis Scale Existence

'NEWSCALE'
'OLDSCALE'

Segment Pickability

'DESENSITIZE'
'SENSITIZE'

Axis Existence

'XYAXES' 'NOAXES'
'XAXIS' 'XYZAXES'
'YAXIS' 'XZAXES'
'YZAXES' 'ZAXIS'

Axis Positioning

'EDGEAXIS'
'ZEROAXIS'
'PENAXIS'
'XEDGEYZEROAXIS' 'YEDGEXZEROAXIS'
'XZEROYEDGEAXIS' 'YZEROXEDGEAXIS'

Numeric Labels

'BESTFORMAT'
'EFORMAT'
'FORMAT'
'GFORMAT'

Label Positioning

'NEGATIVESIDE' 'PERPENDICULARLABELS'
'PARALLELLABELS' 'POSITIVESIDE'
'XYPLANE' 'XZPLANE'
'YPLANE'

Axis Type

'NOLOGARITHMS' 'XYLOGARITHM'
'XLOGARITHM' 'XZLOGARITHM'

'YLOGARITHM'
'YZLOGARITHM'
'ZLOGARITHM'
'XYZLOGARITHM'
'LOGARITHMS'

X Axis Labels

'XNUMERIC'
'XALPHANUMERIC'
'XBOTHLABELS'
'NOXLABEL'

Y Axis Labels

'YNUMERIC'
'YALPHANUMERIC'
'NOYLABEL'
'YBOTHLABELS'

Z Axis Label

'ZNUMERIC'
'ZALPHANUMERIC'
'NOZLABEL'
'ZBOTHLABELS'

Axis Type

'TICAXIS'
'PLAINAXIS'
'GRIDAXIS'

Three Dimensional Viewporting

'SITEPOINT'
'VIEWPOINT'

Windowing

'CWINDOWING'
'NOWINDOWING'
'RWINDOWINE'

Text Output

'NOCENTER'
'ACENTER'
'NOSCRIPT'
'XYZCOORDINATES'
'XYCOORDINATES'
'2DCOORDINATES'
'SUBSCRIPT'
'NOSUBSCRIPTING'
'SUPERSCRIPT'
'INTEGER'
'TEXT'
'3DCOORDINATES'
'HORIZONTAL'
'UPPERCASE'
'VERTICAL'
'REAL'

Menuing Option

'STANDARDMENU'
'EXTENDEDMENU'

Origin Inclusion

'NOORIGIN'
'ORIGIN
'PENORIGIN

Device Space Clipping

'NOZCLIPPING'
'ZCLIP'

High-Level Plotting Option

'NONUNIFORM'
'UNIFORM'

Coordinate Repeat Option

'NOREPEAT'	'NOZREPEAT'	'YCONSTANT'	'ZREPEAT'
'NOXREPEAT'	'XCONSTANT'	'YREPEAT'	
'NOYREPEAT'	'XREPEAT'	'ZCONSTANT'	

Transformation Type

'PERSPECTIVE'
'ORTHOGONAL'

Axis Orientation

'XNEGATIVE'	'YNEGATIVE'	'ZNEGATIVE'
'XPOSITIVE	'YPOSITIVE'	**'ZPOSITIVE'**

Three Dimension Viewing

'XYVIEW'	'XZVIEW'
'XYZVIEW'	'YZVIEW'

Rotation Application Order

'XYZ'	'YZX'
'XZY'	**'ZXY'**
'YXZ'	'ZYX'

Time Axis Scaling

12Hour	'YEARS'	**'DATE'**	'SECONDS'
13Week	'WEEKDAYS'	'MINUTES'	'QUARTERS'
24Hour	'DAYS'	'HOURS'	'PERIODIC'

APPENDIX F

UQUERY OPTIONS

Type	Query Name		Returns (Default In Parentheses)
USET	'ABUTTING'		Page abutting mode ('NONABUTTING')
USET	'ACENTERING'		Alphanumeric centering ('NOCENTERING')
USET	'ADJUSTMENT'	3D	Axis view adjustment option (plane axis plotted on view port or viewed from current view point) ('XYZVIEW')
USET	'ANGULARUNITS'		Current user angular units ('DEGREES')
USET	'ANGLE OF TEXT'		Current angle of text output (0.)
UPSET	'ASPECTRATIO'	3D	Display Surface aspect ratio
UPSET	'ATTENTION QUEUE SIZE'		Attention queue size (100.)
UPSET	'BACKGROUND COLOR'		Background color index (Black=0.)
USET	'BLENDMODE'*		Color Blending mode ('SUBTRACTIVE')
USET	'BLINKRATE'		Blink rate ('NOBLINK')
UPSET	'BRIGHTNESS'		Display intensity (60%)
USET	'BUILD'	3D	Structure build mode flag ('NOBUILD')
UPSET	'CHARACTER'		Current system character as Hollerith string (*)
USET	'CLIPPING'		Device space clipping flag ('NOCLIP', 'CLIP' or 'INVERTED')
USET	'COLOR'		Current color index (device dependent)
UPSET	'COPY DELAY'		Copy delay time (device dependent)
UPSET	'CSPACING'		Character spacing mode ('HORIZONTAL')
USET	'CURVE'		Current curve approximation mode ('CONTINUOUS')
UPSET	'DASH'		Numeric value corresponding to current dashline specification (56.)
USET	'DESCRIPTION'		Current axis option ('TICAXES')
USET	'DETECTABILITY'		Detectability of new segments ('DESENSITIZED')
USET	'DIMENSION'	3D	2D or 3D coordinate terminator switch ('2DCOORDINATES')
UPSET	'DISTANCE'	3D	Distance from viewport to screen plane (150.)
USET	'EDIT'	3D	Data structure edit mode ('COMPRESS')
USET	'ERRORMODE'		Error presentation (ERROR)
USET	'EXECUTE'	3D	Structure building visability ('EXECUTE')
USET	'EXISTENCE'		Current axis existence option ('XYZAXES')
UPSET	'FACTOR'		Scale factor for GCS created software symbols (1.)
USET	'FITMODE'		Curve fitting mode for autoplotting ('NOFITTING')

USET	'FNAMING MODE'		Frame naming mode ('FNAME')
UPSET	'FNTFILE NUMBER'*		Font file number (0.)
UPSET	'FONT NAME'*		Font name ('GCS')
USET	'FORMAT'		Text number format for numeric labelling ('BESTFORMAT')
USET	'GAPMODE'		Gapped line mode ('UINTERRUPTED')
UPSET	'GREYSCALE'		Numeric valve indicating current grey level (device dependent)
UPSET	'GRID'	3D	Numeric value indicating grid axis type option (0.)
USET	'HANDEDNESS'	3D	Left or right handed coordinate system ('RIGHTHANDED')
UPSET	'HARDWARE'		Current hardware character size ('SMALL', device dependent)
UPSET	'HORIZONTAL'		Current horizontal software character position size (5.)
UPSET	'INFILE'		Graphics input file designation (computer system dependent)
USET	'INPUT'		Graphics input device medium
USET	'ITALICIZATION'		Italicization mode ('NOITALICS')
UPSET	'LABELANGLE'		Label angle around perpendicular (0.)
USET	'LETTERTYPE'		Character type ('HARDWARE')
UPSET	'LEVEL'	3D	Subscript/superscript spacing level (1.)
UPSET	'LIBRARY'	3D	Data structure library file mode (0.)
UPSET	'LIMIT'	3D	Number of errors before automatic stop. 0 means no limit (0.)
USET	'LINEOPTION'		Current line option setting ('LNULL')
UPSET	'LOWERCASE'		Lowercase shift character as Hollerith string ('>')
USET	'LOGARITHMIC'	3D	Axes selected for logarithmic transform (logarithmic transform switch) ('NOLOGSCALING')
USET	'LOGTIME OF APPLICATION'		Time of application of logarithmic scaling ('LOGSYSTEM COORDINATES')
USET	'LOGTYPE'	3D	Flag indicating when logarithmic scaling performed. ('LOGSYSTEM', 'LOGUSER', 'LOGOBJECT')
USET	'MAPPINGTYPE'	3D	3D to 2D mapping type ('PERSPECTIVE')
USET	'MENUTYPE'	3D	Menu board type ('STANDARD')
USET	'MERGE'	3D	Structure merge mode switch ('IGNORE')
USET	'MESSAGEDEVICE'		Destination of alphanumeric I/O ('PLOTDEVICE')
USET	'MODE'		Current coordinate mode ('ABSOLUTE')
USET	'NUMERIC'	3D	Numeric labels parallel or perpendicular to axis ('PARALLEL')
USET	'ORDER'	3D	3D coordinate system rotation application order ('ZYX')

UPSET	'ORIENTATION'		Angular orientation of display of GCS created symbols (0.)
USET	'ORIGIN'		Forced origin switch for axis scaling ('ORIGIN')
UPSET	'OUTFILE'		File number of graphics output file (computer system dependent)
USET	'OUTPUTDEVICE'		Graphical output destination device ('ALLDEVICES')
USET	'PLANE'	3D	Axis label plane ('XYPLANE')
USET	'PLOTSCALE'		Scale option ('NEWSCALE')
UPSET	'POLYNOMIALDEGREE'		Current degree of polynominal fit for curve fitting (5.)
USET	'POSITION'	3D	Current axes positioning ('EDGEAXES')
UPSET	'PRECISION'		Number of digits of precision to be displayed for real numbers. (4.)
USET	'PROJECTION TYPE'		Type of projection ('PERSEPCTIVE')
UPSET	'READ FILE'		File designator for non-graphic input (computer system dependent)
USET	'REPEAT'	3D	Data structure invocation option ('SINGLE' or 'REPEAT')
USET	'REWIND'	3D	Structure file rewind mode ('REWIND')
USET	'SCALE'		Axis scale option ('AUTOSCALE')
USET	'SCRIPT'	3D	Subscript/superscript control ('NOSCRIPT')
USET	'SECURITY LEVEL'		Control of security banners ('UNSECURED')
USET	'SENSITIVITY'	3D	Light pen sensitivity switch ('DESENSITIZE')
USET	'SIDE'	3D	Side of axes on which labels will appear ('NEGATIVE')
USET	'SIZE'		Hardware character size ('SMALL')
UPSET	'SLANT'	3D	Software character italic slant angle (18. degrees)
USET	'SPACE'		Coordinate space ('VIRTUAL')
UPSET	'SPAN'	3D	Angular span for pie charts (360. degrees)
UPSET	'START'	3D	Pie chart starting angle (0.)
USET	'STOP'		Error stopping control ('NOABORT')
USET	'STORAGEMODE'		Segment/Frame retention mode ('RETAINED')
UPSET	'STRUCTURE LIMIT'		Maximum number of structures which can be defined (100.)
UPSET	'SUBSCRIPTCHARACTER'	3D	Current subscript shift character
UPSET	'SUPERSCRIPTCHARACTER'	3D	Current superscript shift character
UPSET	'SYMBOL INDEX'		Choice of marker (0.)
USET	'SYSTEM'		Current coordinate system ('SYSTEM')
UPSET	'TABHORIZONTAL'		Location of current horizontal tab stop (10.)
UPSET	'TABVERTICAL'		Location of current vertical tab stop (10.)
UPSET	'TERMINATOR'		GCS string termination character

USET	'TEXT'		Textual I/O mode ('TEXT')
UPSET	'TICINTERVAL'		Length of UPEN ticintervals (10.)
UPSET	'TICLENGTH'		Length of UPEN ticintervals (10.)
UPSET	'TICMINUS'		Clockwise tic mark size (0.05)
UPSET	'TICPLUS'		Counter-clockwise tick mark size (0.05)
UPSET	'TICX'		X axis tic interval (0.)
UPSET	'TICY'		Y axis tic interval (0.)
UPSET	'TICZ'	3D	Z axis tic interval (0.)
USET	'TIME'		Time series plotting period ('DATES')
USET	'TYPE'		Type of coordinates ('RECTANGULAR)
USET	'UNIFORM'	3D	High level plotting option ('NONUNIFORM')
USET	'UNITS'		Device space units ('INCHES')
USET	'UPAXIS'	3D	Coordinate system viewport vertical axis ('ZPOSITIVE')
UPSET	'UPPERCASE'		Uppercase shift character as Hollirith string ('<')
USET	'USER'		User coordinate system switch
UPSET	'VERTICAL'	3D	Vertical software character position switch
USET	'VIEWPORT'		Viewport distance base switch ('SITE')
USET	'VISIBILITY'		Framed/segment creation visibility mode ('VISIBLE')
UPSET	'WIDTH OF LINES'	3D	Width of lines (0.)
USET	'WINDOW'		Window type ('NOWINDOW')
UPSET	'WRITE FILE'		Non-graphic alphanumeric output file (Computer system dependent)
UPSET	'XBASE'		Base of log scaling along x-component (real number or string 'E') (10.)
USET	'XLABEL'		X axis labelling option ('XNUMERICLABEL')
USET	'XLOGARITHMIC'		X axis linearity option
UPSET	'XPERCENTAGE OF CHARACTER SPACE'		Portion of horizontal character space occupied by character (0.65)
USET	'XREPETITION MODE'		X component repetition mode for plotting ('NOXREPEAT')
UPSET	'XROTATION'		Rotation factor around X axis for structure invocation (0.)
UPSET	'XSCALING'		Scaling factor along X axis for structure invocation (1.)
UPSET	'XSPECIFICATION UNIT'		Number of XSPECIFICATION unts in device space (1000.)
USET	'XSIZE'		Horizontal hardware character position size (device dependent)
UPSET	'XTITLE'		X axis alphanumeric label ('X')

UPSET	'YBASE'	3D	Base of log scaling along Y-component (real number or string 'E') (10.)
USET	'YLABEL'		Y axis labelling option ('YNUMERICLABELS')
USET	'YLOGARITHMIC'		Y axis linearity option
UPSET	'YPERCENTAGE OF CHARACTER SPACE'		Portion of vertical character space occupied by character (0.65)
USET	'YREPETITION'		Y component repetition made for plotting ('NOYREPEAT')
UPSET	'YROTATION'		Rotation factor around Y axis for structure invocation (0.)
UPSET	'YSCALING'		Scaling factor along Y axis for structure invocation (1.)
UPSET	'YSIZE'		Vertical size for hardware characters (device dependent)
UPSET	'YSPECIFICATIONUNITS'		Number of Y specification units in device space (1000.)
UPSET	'YTITLE'		Y axis alphanumeric label
UPSET	'ZBASE'	3D	Base of log scaling along Z axis (10.)
USET	'ZCLIP'		Hither/yon clipping mode ('NOZCLIPPING')
USET	'ZLABEL'	3D	Z axis label option ('ZNUMERICLABELS')
USET	'ZREPETITION MODE'		Z component repetition mode for plotting ('NOZREPEAT')
UPSET	'ZROTATION'		Rotation factor around Z axis for structure invocation (0.)
UPSET	'ZSCALING'		Scaling factor along Z axis for structure invocation (1.)
UPSET	'ZSPECIFICATIONUNITS'		Number of Z specification units in device space (1000.)
UPSET	'ZTITLE'	3D	Z axis title
UPSET	'ZVALUE'	3D	Numeric default Z-value for 2D coordinates (0.)

APPENDIX G

GCS ERROR CODES

The following is a list of the currently defined error codes in GCS:

01 — Invalid key word specification to USET.
02 — Invalid key word specification to UPSET.
03 — No plot files found.
04 — Invalid option for UPRNT1.
05 — Invalid UDOIT system.
06 — Invalid view port boundaries.
07 — UFORMT — Invalid Display Surface Format.
08 — Invoke font specification.
09 — Invalid UINPUT option.
10 — UMARGN argument list out of order/Boundary specification invalid.
11 — UMARGN boundary outside of physical device boundary.
12 — UWINDO argument list out of order/Boundary specifications invalid.
13 — UDAREA argument list out of order/Boundary specifications invalid.
14 — UDAREA boundary outside of physical device boundary.
15 — UCLIP invalid argument list.
16 — UCLIP boundary overlap error.
17 — UDIMEN maximum boundary specification invalid.
18 — UAXIS argument list XMIN .GT.XMAX and/or YMIN .GT. YMAX.
19 — UDAREA provided to UAXIS too small for requested options.
20 — UPSET — invalid tic interval. Value .LE.zero specified.
21 — UPSET — invalid scale factor. Value must not be equal to zero.
22 — UPSET — invalid software character size. Value must not be equal to zero.
23 — UPSET — invalid digits of precision. Precision must be greater than zero.
24 — UPSET — attempt to set zero or negative scripting — level.
25 — UPSET — Invalid light pen correlation value. Value must be.GT.zero.
26 — UPSET — Invalid transmission speed. Value must be .GT.zero.
27 — UPSET — Invalid Library file code. File codes must be in range 1 to 99.
28 — UPSET — Invalid error limit. Value must be greater than zero.
29 — UPSET — Invalid error limit. Value must be greater than zero.
30 — UFRAME/UFREND frame table full.
31 — Failure to call UFREND for previous occurrence of same frame.
32 — Failure to call UFREND for another named frame.
33 — UFREND called without any frame active.
34 — UFREND call not for currently active frame.
35 — USHOW/UNSHOW called for undefined frame.
36 — USHOW/UNSHOW called while in frame build status.
50 — PLOT — Constant or delta X or Y values illegal for automatic curve fitting.
51 — Insufficient data points for desired fit.
52 — Input data points exceed fit capacity.
53 — Input data out of order or non functional relationship.
54 — Insufficient array space to return fitted results.
55 — Invalid trend adjustment factor.
56 — Invalid polynomial degree for polynomial fit.
57 — Invalid number of previous periods for moving average fit.
60 — Attempt to build a structure in frame mode.
61 — Nested structure call stack overflow.
64 — Attempt to redefine existing structure.
65 — Structure table overflow.
66 — Attempt to activate a structure while another is still active.
67 — Structure termination without active structure.
68 — Structure termination not for current structure.

69 — Attempt to execute an undefined structure.
70 — Recursive structure build call.
71 — Invalid number of items for UREAD/UINPUT.
72 — Attempted input operation from batch device.
80 — Attempt to create a secondary axis scale of zero.
81 — UAXIS — pen position outside of UDAREA in 'PENAXES'.
82 — Attempt to plot log with window bound .LE. 0.0
83 — Attempt to move to apply log scaling to coordinate whose value is zero.
84 — Log plotting in device space not allowed.
85 — UAXIS — AXIS choice requires 0 to 1 within range of X and/or Y AXIS.
86 — ULINE/U3LINE — Invalid number of points.
91 — UAPEND found zero length string. Terminator placed in first character position.
92 — UWAIT — Negative time period specified.
93 — UQUERY — Invalid option specification.
94 — UCOLOR — Invalid color index.
95 — UCOLOR — Duplicate color component.
96 — UCOLOR — Duplicate color component.
97 — UCOLOR — Invalid color name.
98 — UCOLOR — Invalid color component value.
99 — UCOLOR — Colinear with line.
101 — UHISTO — Insufficient points.
102 — UHISTO — Invlid number of bars.
103 — UHISTO — Unreasonable window for OWNSCALE.
104 — UHISTO — Insufficient UDAREA for options specified.
110 — Invalid number of points for UPIE.
111 — Invalid data value for UPIE.
112 — Invalid max label size for UPIE.
113 — Insufficient room for UPIE display.
114 — UPIE — ABS(Starting Angle > = ABS (Ending Angle).
115 — UPIE — Too many labels outside of pie.
120 — USCATR — Insufficient points specified.
121 — USCATR — Invalid limits for logarithmic scatter diagram.
122 — USCATR — UDAREA too small for specified options.
130 — UBAR — Invalid number of points.
131 — UBAR — Invalid label size.
132 — UBAR — Invalid data value.
133 — UBAR — Insufficient UDAREA for options specified.
140 — UCHART — Invalid number of points.
141 — UCHART — Invalid label size.
142 — UCHART — Invalid number of bars.
143 — UCHART — UDAREA too small for specified options.
150 — UTAXIS — UDAREA too small for specified options.
160 — UTILTY — Invalid action.
161 — UTILTY — Save structure not found.
162 — UTILTY — Utility operation on structure work file.
164 — UREPRO — Invalid pseudo-device file number.
165 — UREPRO — Invalid pseudo-device file format.
166 — UREPRO — Read error on pseudo-device file.
167 — UREPRO — Invalid pseudo-device command.
168 — UREPRO — End-of-file within pseudo-device command.
169 — UREPRO — Open segment of UREPRO invocation.
190 — UVIEW — Viewpoint specified same as view site.
191 — UVWPRT — Aperature specified negative or zero in some dimension.
192 — UVWPRT — Viewport behind viewer.
193 — Attempt to draw through viewpoint.
194 — Hither plane behind or at viewpoint.

☆U.S. GOVERNMENT PRINTING OFFICE 1980-640-289/39

In accordance with letter from DAEN-RDC, DAEN-ASI dated 22 July 1977, Subject: Facsimile Catalog Cards for Laboratory Technical Publications, a facsimile catalog card in Library of Congress MARC format is reproduced below.

Westinghouse Word Processing Center, Pittsburgh.
Primer on computer graphics programming / by Westinghouse Word Processing Center, Pittsburgh, Pa. Vicksburg, Miss. : U. S. Waterways Experiment Station ; Springfield, Va. : available from National Technical Information Service, 1979.
v, [201] p. : ill. ; 27 cm. (Miscellaneous paper - U. S. Army Engineer Waterways Experiment Station ; O-79-4)
Prepared for Office, Chief of Engineers, U. S. Army, Washington, D. C., under Contract No. DACW39-78-M-2676.

1. Computer graphics. 2. Computer programming. 3. Graphics Compatibility System. I. United States. Army. Corps of Engineers. II. Series: United States. Waterways Experiment Station, Vicksburg, Miss. Miscellaneous paper ; O-79-4.
TA7.W34m no.O-79-4

WATERWAYS EXPERIMENT STATION REPORTS PUBLISHED ON THE GRAPHICS COMPATIBILITY SYSTEM

	Title	Date
Miscellaneous Paper 0-79-1	Host-Computer Implementation Guidelines for the Three-Dimensional Graphics Compatibility System (GCS)	Mar 1979
Miscellaneous Paper 0-79-2	Device Implementation Guidelines for the the Three-Dimensional Graphics Compatibility System (GCS)	Mar 1979
Miscellaneous Paper 0-79-3	Raster Graphics Extensions to the Graphics Compatibility System (GCS)	Mar 1979
Miscellaneous Paper 0-79-4	Primer on Computer Graphics Programming	Oct 1979
Miscellaneous Paper 0-79-5	Graphics Compatibility System (GCS) Programmer's Reference Manual	Oct 1979